INFORMING SPACECRAFT MANEUVER DECISIONS TO REDUCE PROBABILITY OF COLLISION

THESIS

Elizabeth-Ann R DeNeve, Captain, USAF

AFIT-ENY-14-M-15

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENY-14-M-15

INFORMING SPACECRAFT MANEUVER DECISIONS TO REDUCE PROBABILITY OF COLLISION

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Astronautical Engineering

Elizabeth-Ann R DeNeve, BS

Captain, USAF

March 2014

DISTRIBUTION STATEMENT A.
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENY-14-M-15

INFORMING SPACECRAFT MANEUVER DECISIONS TO REDUCE PROBABILITY OF COLLISION

Elizabeth-Ann R DeNeve, BS

Captain, USAF

Approved:

____//signed//_________________________ 5 March 2014
William Wiesel, PhD (Chairman) Date

____//signed//_________________________ 5 March 2014
Ronald Simmons, LtCol, USAF (Member) Date

____//signed//_________________________ 5 March 2014
Richard Erwin, PhD (Member) Date

AFIT-ENY-14-M-15

Abstract

Space is becoming increasingly congested as more objects are launched into orbit. The potential for a collision on orbit increases each time a new object enters space. This thesis presents a methodology to determine an optimal direction to maneuver a satellite that may be involved in a potential collision. The author presents a paradigm to determine the optimal direction of maneuver to achieve the lowest probability of collision, and examines how different magnitudes of a maneuver will affect the probability of collision. The methodology shows that if a satellite maneuvers in the optimal direction at any time during the orbit, except incremental periods and half periods, the probability of collision is reduced to a negligible amount. This provides a means to determine a maneuver direction and magnitude that will remove satellites from the potential collision area, while minimizing the resources necessary and maintaining mission quality.

Table of Contents

List of Figures

Page

List of Tables

Page

INFORMING SPACECRAFT MANEUVER DECISIONS TO REDUCE PROBABILITY OF COLLISION

I. Introduction

General Issue

Satellite collisions have become a more prominent topic in the space community. The first man-made object was launched into orbit in 1957, Sputnik 1. Since then there has been a rapid and dramatic increase in the number of objects in orbit. Until recently, there was little concern over the possibility of a collision between two objects, once considered a one-in-a-million chance. To put this in perspective, one radial degree covers approximately 14,400 square kilometers in a standard low Earth orbit. This is ten times the size of Los Angeles, California. At any time there could be hundreds of objects within that area with no real chance of collision. Two objects are considered within collision potential on orbit when they are separated by a mere two kilometers. For two objects in orbit, two kilometers is dangerously close and there is increased concern at the thought of a close approach. The most recent incident of a satellite collision was the Iridium 33-Cosmos 2251 collision in 2009. Not only was the multimillion dollar Iridium satellite lost, the Iridium-Cosmos collision created nearly 1,000 space debris objects in each satellite's orbit (Satellite Collision Leaves Significant Debris Cloud, 2009). The increase in the debris count after each collision increases the probability that more collisions will occur in the future. Every time a new object is placed in orbit, be it an active satellite that just launched, the rocket body it arrived on, the dead satellite it is replacing, or the pieces from previous collisions, each object must be tracked to ensure the safety of our satellite constellations. There are approximately 21,000 objects in orbit that can be actively tracked by ground-based radar systems and an even larger number of objects that are too small to track consistently (NASA Orbital Debris Program Office, 2012). Figure 1 below shows the increase in the amount of catalogued objects since 1957 (Orbital Debris Quarterly News, 2011). The graph below depicts growth of catalogued objects based on real events; the expected outcome is that as more debris is introduced the amount of objects will increase exponentially (Orbital Debris Quarterly News, 2011).

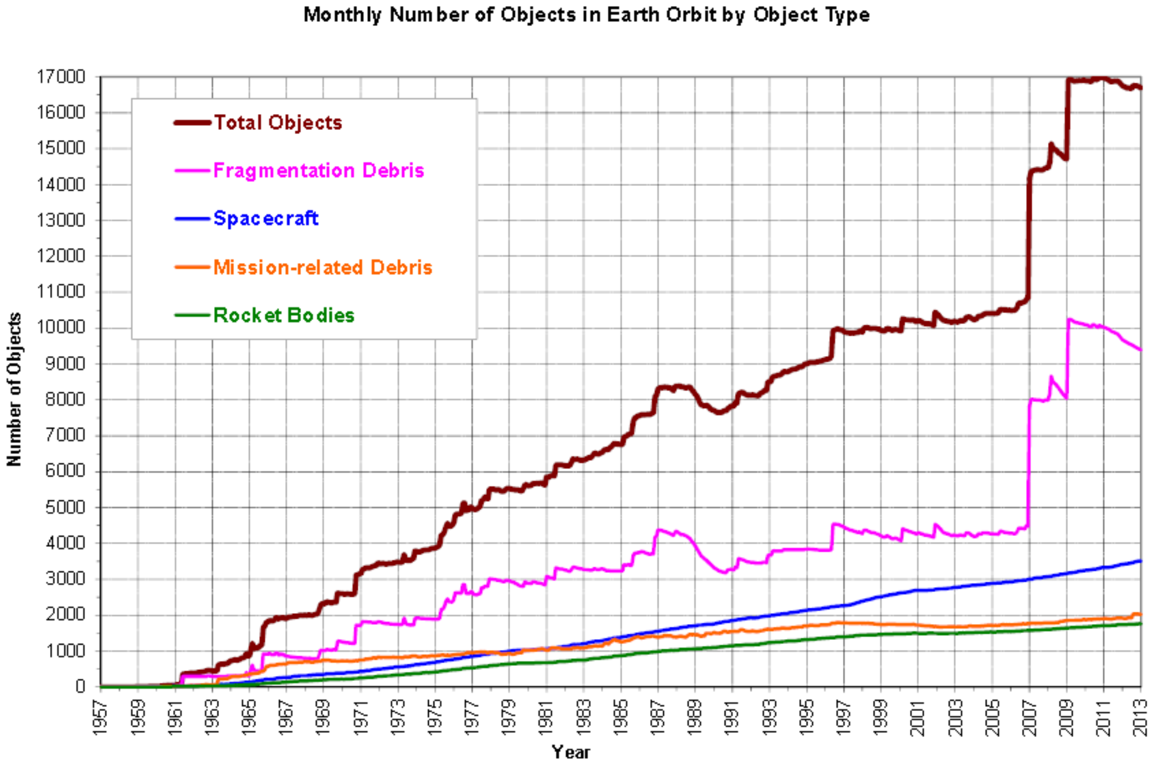


Figure 1: Yearly catalogue of objects in Earth orbit by object type.

The United States Space Surveillance Network (SSN), a division of the Joint Space Operations Center (JSpOC), is responsible for detecting, tracking, cataloguing, and identifying all man-made objects in Earth’s orbit (USSTRATCOM Space Control and Space Surveillance, 2013). The SSN uses multiple radar and optical sites to observe objects in orbit, daily. The JSpOC uses the information collected from the SSN to predict the future position of objects. With these data, the JSpOC can identify potential future collisions; this process is known as conjunction analysis.

A conjunction is when two objects collide in orbit. Most analysis of potential conjunctions assumes perfect knowledge of each object’s orbit involved in the conjunction; however, there is always some amount of uncertainty in every object’s dynamic state. The uncertainty of each object’s position varies widely from object to object. Due to this uncertainty, there is a rising concern that an increased number of object collisions will occur in the near future. The uncertainty is important since perfect position and velocity is not known, the object could be anywhere within the uncertainty. This causes problems when the uncertainties of two objects overlap. Since each object could be anywhere within that uncertainty, the potential to collide increases. For example, a collision scenario could produce a miss distance of 500 meters, but due to the uncertainty in the object’s position, the distance between the two objects could be two kilometers away or zero meters (a collision). The only way to improve this situation is to reduce the amount of uncertainty surrounding the position of each object. By reducing the amount of uncertainty, knowledge of the object’s position is better known and the potential for the two possible positions to overlap is reduced. However, the increasing number of objects in orbit make keeping track of objects more difficult.

The Air Force Space Command (AFSPC) conjunction message provides enough information about the object’s predicted state to calculate a current collision probability estimate. Additionally, the message includes sufficient information to calculate a change in the collision probability if an object contemplates a maneuver. When JSpOC identifies a potential collision, each of the satellite owners are notified of the potential conjunction via a Conjunction Summary Message (CSM). The JSpOC produces CSMs daily and a satellite owner is informed of inclusion in a potential collision up to seven days in advance (USSTRATCOM Space Control and Space Surveillance, 2013). The CSM provides the date the report was created, date and time of the conjunction, potential miss distance, position and velocity vectors for both objects, and the uncertainty of the object’s position, velocity, and each object’s covariance (USSTRATCOM Space Control and Space Surveillance, 2013).

A CSM identifies both objects involved in a potential collision. Assuming at least one of the satellites has the capability to maneuver; the conjunction message will be generated and distributed to satellite owners. Assuming the satellite has the capability to maneuver, then a maneuver may be conducted in attempts to reduce the probability of collision. The maneuver decision is solely up to the satellite owner. The JSpOC has no authority to order satellite owners to maneuver their satellites. They provide the predicted miss distance and the uncertainty from which a satellite owner can calculate a probability of collision. The JSpOC does not provide a calculated probability of collision, only the information that could be used to generate that value.

The satellite operator typically has enough time to command a maneuver to the spacecraft to try to avoid the collision. However, fuel is a precious commodity on a spacecraft. The amount of fuel that a satellite has varies based on satellite size and mission. Some satellites carry enough fuel to perform many maneuvers over the lifetime of the satellite, whereas others carry only enough to make a few or no maneuvers at all. Currently, it is not financially practical to launch a refuel mission to an on-orbit asset. The common practice is to launch a new satellite to replace the one that can no longer fulfill mission needs. For this reason, if you run out of fuel, normally your mission will end. While most spacecraft have the ability to maneuver, such maneuvers can negatively affect the spacecraft's mission. A maneuver may disrupt the calibration of the spacecraft and time will be lost attaining spacecraft functionality again. Or, the maneuver could move the spacecraft away from its intended target, requiring it to perform an additional maneuver to get back to its mission orbit, or be re-purposed. All of these situations could be detrimental to the mission.

For these reasons, it is crucial to develop a method to maneuver an object so probability of collision is reduced is crucial to the growing number of satellite conjunctions. A method to avoid conjunctions would give satellite owners additional information about the projected conjunction so they can make an informed decision on corrective action.

Problem Statement

The lack of accurate orbit knowledge, the growing number of objects in orbit, and the lack of the necessary information needed to make an informed maneuver decision lead to the potential for more satellite collisions. Currently there is no methodology that exists to specify maneuver times and magnitudes to reduce probabilities of collision while minimizing fuel expenditure and effects on mission parameters.

Research Goals

This thesis is focused on determining how a satellite's collision probability can be minimized by performing a collision avoidance maneuver. This research determines three essential elements for maneuver times and magnitudes to reduce probability of collision and impacts to orbital parameters.

1. Estimate the probability of collision changes from different magnitudes of maneuver
2. Determine optimal maneuver direction and maneuver times to reduce probability of collision
3. Determine optimal maneuver direction to reduce the impact on certain orbital parameters while reducing probability of collision

Scope

The primary purpose of this thesis is to determine a methodology to calculate a maneuver that avoids the potential collision area. This thesis focuses on the major contributing factors in orbit determination and evaluates the potential to reduce collisions. One collision scenario is developed and used for the entire study to determine effects on probability of collision of various maneuvers performed. Both objects involved in the scenario are modeled as point masses and maneuvers performed by an object are impulsive.

II. Literature Review

Chapter Overview

The purpose of this chapter is to describe the necessary background information to support the methodology, research, and conclusions in future chapters. The topics covered in this chapter are a discussion of the current probability of collision models, the satellite orbital parameter propagator, and the effects of maneuvering satellites.

Probability of Collision

With the increase in the number of Earth-orbiting objects, there is a growing concern that more objects will collide with one another. There are three main types of objects orbiting Earth: active satellites, inactive satellites, and debris. Active satellites are currently being operated by a user and are currently used for mission operations. Inactive satellites are formerly active satellites that have been disabled or shut down and no longer function. Debris is everything else; rocket bodies, fragments of satellites, and so on (USSTRATCOM Space Control and Space Surveillance, 2013). The likelihood that any two objects (across all three categories) will collide is defined as the probability of collision. The JSpOC is primarily concerned with active satellite to other object encounters. This includes active to active, active to inactive, or active to debris (USSTRATCOM Space Control and Space Surveillance, 2013). In these situations, most likely something can be done to avoid the collision. The probability that a collision will occur is defined by a three dimensional Gaussian probability density function defined as:

$$P_c = \frac{1}{\sqrt{(2\pi)^3|C|}} \iiint\limits_V e^{-\frac{1}{2}\vec{r}^T C^{-1}\vec{r}} dXdYdZ \tag{1}$$

Where P_C, the probability of collision "is defined as the integral of the probability density function over the swept out volume V," and "C is defined as the sum of the primary's and secondary's position covariance matrices" (McKinley, 2006: 2). The equation above is the general Gaussian function to describe the probability of collision. The integral is difficult to evaluate since the combined covariance matrix, C, of the two objects is changing in time as each object moves along its trajectory (McKinley, 2006). Many researchers have evaluated this function and with simplifying assumptions, the probability of collision can be calculated.

One such assumption is that the relative motion between the primary and secondary objects is rectilinear (McKinley, 2006). This assumption allows for the above equation to be reduced to a two dimensional integral. The rectilinear assumption is well-known and used by multiple researchers (Chan, 1997; Patera, 2001; Berend, 1999). The two dimensional reduction of the above equation is

$$P_c = \frac{1}{\sqrt{(2\pi)^2|C^*|}} \iint\limits_A e^{-\frac{1}{2}\vec{r}^T C^{*-1}\vec{r}} dXdZ \tag{2}$$

Where, "C^* is the two dimensional covariance matrix that results from projecting C into the conjunction plane" (McKinley, 2006: 3). The two-dimensional version of the multivariate Gaussian distribution is more commonly used; other researchers have a slightly different version based on their definition of certain parameters and the reference frame used. According to multiple researchers (McKinley, 2006; DeMars et al., 2014;

Patera, 2001), the above simplifying assumption can only be made under certain circumstances. The two dimensional probability density function is only valid for encounters of short duration; this implies a high relative velocity between the two objects and that the covariance matrices of each object are constant over a short duration (McKinley, 2006; DeMars et al., 2014; Patera 2001).

A covariance matrix provides information about how one variable has statistical dependence on another variable. In this simulation, the covariance matrix used for calculation is a combination of the covariance matrix from each object. When this combined covariance matrix is statistically independent it means the position and velocity of one object have no bearing on the position and velocity of the second object. The values on the diagonal are called variances and are the standard deviation squared (σ_{ii}^2) quantities (Wiesel, 2010A). The off-diagonal values are the covariance quantities (Wiesel, 2010A). When the covariance values are equal to zero, the covariance matrix becomes diagonal, it can then be modeled as the product of a one dimensional Gaussian (Wiesel, 2010A). If the covariance matrix is diagonal, this shows that each variable is statistically independent (Wiesel, 2010A). Additional research has studied the impacts of different covariance shapes. A survey of research articles about calculating collision probabilities shows that most methods for calculating the covariance matrix begin with a spherical shape. An orbiting object increases its uncertainty every time a maneuver is performed. This starts back with the launch trajectory. The launch vehicle places the object approximately at the drop off location, the object then performs maneuvers to reach the desired orbital placement. The object starts with the uncertainty of the launch vehicle and then adds additional uncertainty for every maneuver performed after. An orbiting object decreases its uncertainty when a tracking method locates its position. When a maneuver is performed, the additional uncertainty in the object can only be reduced with tracking. Tracking objects more frequently will result in better knowledge of the object's location, reducing the uncertainty. Objects that are not tracked as often still maintain some degree of uncertainty, variance, in their position and velocity (Wiesel, 2010A). The figure below depicts two objects and their uncertainties, *S,* at the potential collision time.

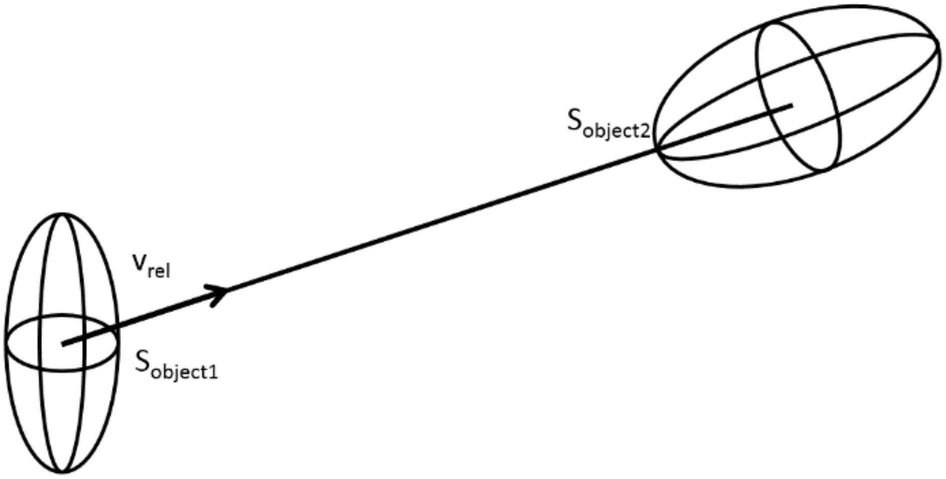


Figure 2: Covariance Geometry

The uncertainty of each object can be modeled by its covariance. Methods for calculating the covariance matrix include spherical, elliptical and more detailed object specific shapes. For the scope of this problem, spherical covariance matrices are used. The research performed on a spherical covariance shape assumes the probability density function is constant over the entire sphere and the uncertainty in each object's velocity was not included since the uncertainty in each object's velocity is irrelevant at the conjunction time (Alfriend et al., 1999). The uncertainty in each object's velocity is not considered because the calculations for determining the probability of collision do not rely on the velocity at the close approach time. A simplifying technique combines the position covariance matrices for both objects to create a relative position covariance matrix for the close approach (Wiesel, 2010A). The combined covariance matrix is dependent on the relative position vector between the two objects. The combined covariance is:

$$S_{rel} = E\left(\left(\delta r_{object1} - \delta r_{object2}\right)\left(\delta r_{object1} - \delta r_{object2}\right)^{T}\right) \qquad (3)$$

Where $\delta r_{object1}$ is the change in the position of object 1, $\delta r_{object2}$ is the change in position of object 2, and E is the expected value operator (Wiesel, 2010A). The combined covariance matrix is calculated by taking the expected value of the relative position vector. The equation can be simplified because this estimation operator is a linear operator:

$$S_{rel} = S_{object1} + S_{object2} \qquad (4)$$

Where $S_{object1}$ is the covariance matrix for object 1, $S_{object2}$ is the covariance matrix for object 2, and S_{rel} is the combined covariance matrix of the two objects (Wiesel, 2010A). This simplification can be made if it is assumed that the position vectors of the two objects are statistically independent (Wiesel, 2010A).

This combined covariance matrix has a “three-dimensional probability density function that represents the uncertainty in relative position between the two objects,” (Patera, 2001: 716). This three-dimensional probability density function can then be reduced as stated in the paragraph above to a two-dimensional probability density function for ease of calculation (Patera, 2001). This combined covariance matrix contains the uncertainty of both orbiting objects and is used for the remainder of this study.

Simplified General Perturbations, SGP4

The SGP4 propagator is a dynamic orbital propagator using the classical set of orbital elements for propagation. The version of SGP4 for this study uses the set of equinoctial orbital elements for propagating. This in-house modification was made to avoid most singularities that occur with classical orbital elements. The equinoctial elements provide orbital parameter data when used with singular orbits where classical elements fail (Vallado and Crawford, 2008). The seven components of the equinoctial element set are mean motion (a classical element), mean longitude, drag coefficient (constant), h, k, χ, and ψ. The following equations show the calculations for the equinoctial elements based on the classical set (Vallado and Crawford, 2008):

$$mean\ longitude = M + \omega + \Omega \tag{5}$$

$$h = e\sin(\omega) \tag{6}$$

$$k = e\cos(\omega) \tag{7}$$

$$\chi = \tan\left(\frac{i}{2}\right)\sin(\Omega) \tag{8}$$

$$\psi = \tan\left(\frac{i}{2}\right)\cos(\Omega) \tag{9}$$

Where M is the mean motion, ω is the argument of perigee, Ω is the right ascension of the ascending node, and i is the inclination. The SGP4 propagator code provides a function to convert the classical set to the equinoctial set. The SGP4 model predicts orbital parameters based on orbital perturbations: Earth’s non-spherical nature, atmospheric drag, solar, lunar, other body, and deep space effects. SGP4 takes an initial set of orbital elements, and propagates them forward (or backward) in time to give an estimated set of orbital elements at the specified time. This set of elements can then be converted back to a classical set of orbital elements or the position and velocity vector of the object at that time. This propagating system is useful to determine where an object will be in the future to determine if there is a possibility of a close approach. SGP4 is the model that AFSPC uses to determine potential close approaches for their CSMs (USSTRATCOM Space Control and Space Surveillance, 2013).

Maneuvering Satellites

Maneuvering satellites on orbit affects some of the orbital parameters regardless of the maneuver direction. Depending on which direction the satellite maneuvers different orbital parameters are affected. Individual satellites require different parameters to remain constant based on constellation, revisit time, location accuracy and mission need. For example, the Iridium satellites are in a Walker constellation and each satellite has a specific location that it must maintain. This constellation requires not changing the orbital period of the satellite. The Iridium satellites provide continuous coverage of the Earth; there are 66 satellites in 6 planes spaced out 11 satellites in each plane (Wertz et al., 2011). If one of the satellites moves out of its designated location, this will disrupt the requirement for continuous coverage of the Earth for the Iridium constellation.

First, a look at how maneuvering in the three orthogonal directions according to the satellite coordinate frame affect satellite parameters. The figure below depicts an example of a body-fixed coordinate frame for a satellite. The direction marked “1” below is in the direction of the velocity vector of the satellite. The direction marked “2” below is the radial direction, from the center of the Earth to the satellite. The direction marked “3” below is the final direction to complete the right-handed system.

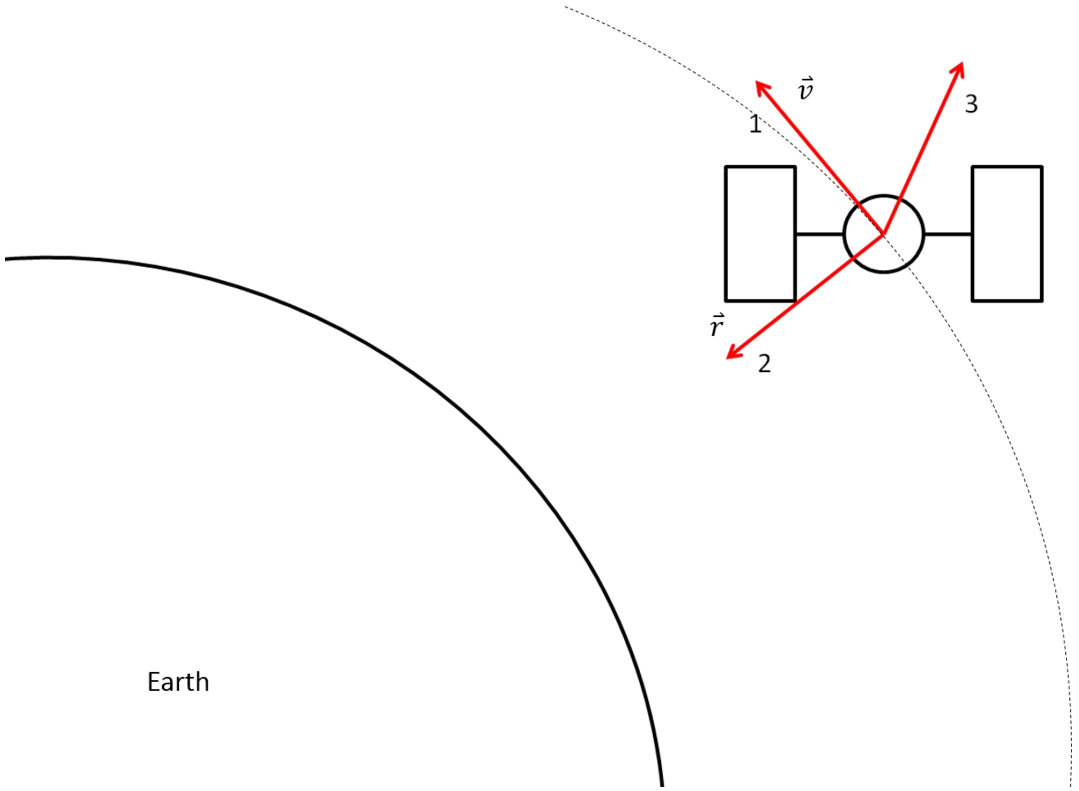


Figure 3: Satellite fixed coordinate frame

A satellite that maneuvers in the direction of the velocity vector, the “1” direction, will change the semi-major axis, eccentricity, period and sometimes the argument of perigee. The argument of perigee will change if the maneuver occurs at some point other than perigee (Wiesel, 2010B). If the satellite maneuvers in the radial direction, the “2” direction, then the same orbital parameters as above will change; semi-major axis, eccentricity, period, and argument of perigee with the same caveat (Wiesel, 2010B). If the maneuver occurs in the final orthogonal direction, the “3” direction, the inclination and argument of perigee will change (Wiesel, 2010B). If a maneuver is performed in a direction other than the orthogonal coordinate frame, you will see a combination of the various effects on your orbital parameters.

Summary

In this chapter, the background information about probability of collision and probability density functions is discussed. Information about the propagation tool and the numerical integration technique utilized is described. Additionally, the effect on orbital parameters from maneuvering satellites is presented. The background information presented supports the methodology, results and conclusions in future chapters.

III. Methodology

Introduction

This chapter describes the methodology used to calculate the probability of collision for a given scenario. First the process of generating an SGP4 scenario to match the projected close approach scenario is explained. Second, the approach for propagating the satellite backwards to the maneuver time is discussed. Next the algorithms designed to calculate the maneuver direction are described. Then, the method used for calculating the probability of collision for each scenario to show how the probability can change based on changing aspects of the maneuver is presented. Finally, the simulation design methodology is described.

Generating the scenario

The scenario is generated for the purpose of this study to ensure a high probability the two objects will collide. It is essential to define some of the terminology that will be used in the remainder of this study.

- Target satellite: The satellite in the scenario that has the capability to maneuver and is the primary satellite.
- Victim satellite: The satellite in the scenario that has no capability to maneuver.
- Close approach time: The time defined where the target and victim are closest. If a collision occurs, this is the time of the collision. This time is also referred to as epoch time.
- Maneuver time: The time chosen prior to the close approach time to perform a change in velocity maneuver.

In practice, a close approach scenario will come from a CSM. The scenario developed starts with a position and velocity vector at the close approach time. These values are chosen so that they are within the uncertainty of each object's position. This ensures the probability that the two objects collide is high. The scenario contains information for both the target and victim satellite to include the satellite number, position vector (in inertial frame), velocity vector (in inertial frame), date and time of close approach, air drag coefficient, and the joint covariance matrix.

Since a maneuver must be performed prior to the close approach time, it is logical to propagate the satellite backwards in time to determine the time to maneuver. The following figure shows the methodology used for propagating a satellite backwards and performing a maneuver to determine a new probability of collision. The simulation begins at the collision time since a CSM provides the position and velocity of both objects at the time of potential collision. To determine how a maneuver affects the location of the target satellite, the target satellite is propagated backwards from the initial conditions. Based on the position of the target and victim satellites at the time of collision, the direction the target satellite maneuvers is determined. The target satellite performs the maneuver at various times and magnitudes to show how the probability of collision changes for each variable. The following figure shows how the initial conditions for each scenario are determined by starting with a collision, then propagating backwards, determining the maneuver, and calculating the probability of collision based on changes in time and magnitude of the maneuver. The processes in this figure are explained in further detail in the following sections.

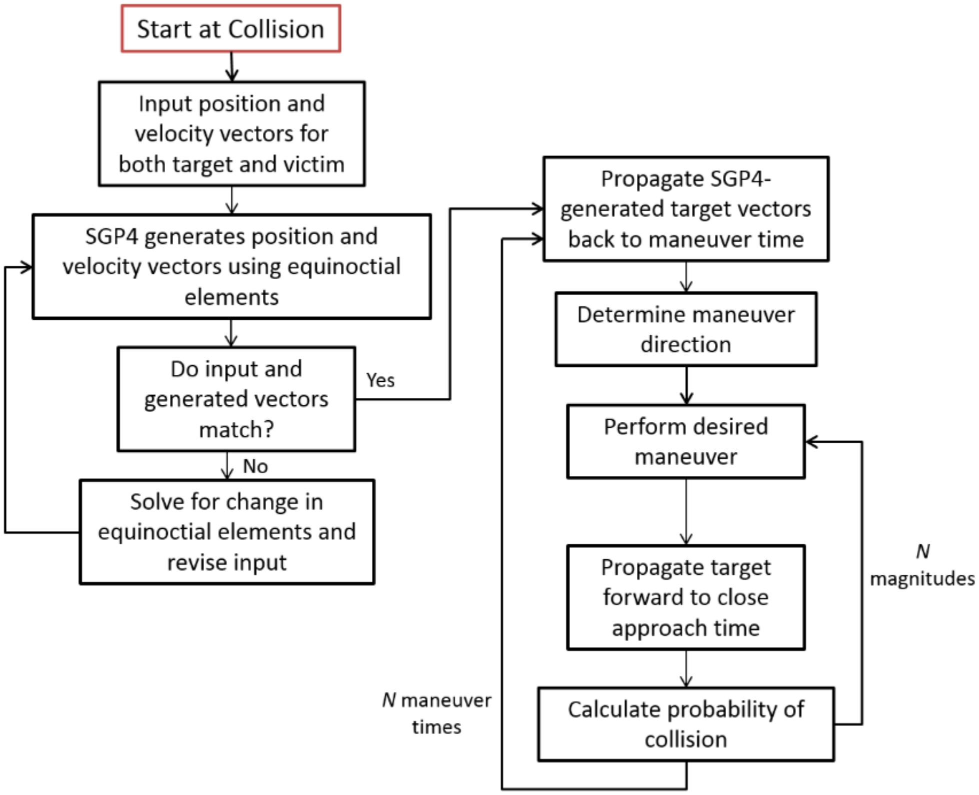

Start at Collision
Input position and velocity vectors for both target and victim
SGP4 generates position and velocity vectors using equinoctial elements
Do input and generated vectors match?
Yes
No
Solve for change in equinoctial elements and revise input
Propagate SGP4-generated target vectors back to maneuver time
Determine maneuver direction
Perform desired maneuver
Propagate target forward to close approach time
Calculate probability of collision
N magnitudes
N maneuver times

Figure 4: Overview of Methodology Flow Chart

Creating SGP4 generated vectors

Since SGP4 utilized equinoctial elements for propagation, the position and velocity vector must be converted to equinoctial elements. This process begins with converting the initial position and velocity vectors to a classical orbital elements set. The SGP4 program contains a function that performs this action. The classical orbital elements must then be converted to the equinoctial element set. The SGP4 program contains a function that will perform this action. A set of equinoctial elements is generated for both the target and the victim satellite from their provided position and velocity vectors given in the conjunction scenario. The initial set of equinoctial elements is inputted into the SGP4 propagating tool and *propagated* to the close approach time.

This process creates an SGP4-generated position and velocity vector at the time of close approach. These vectors are slightly different than what was originally provided due to the perturbing effects that SGP4 has within its processes. The position and velocity vectors that are produced in the SGP4 scenario need to match the provided vectors. The SGP4 generated position and velocity vectors require an iterative process for them to converge on the initially provided vectors. This will provide a set of equinoctial elements that corresponds to the provided position and velocity vectors. To get the change in the equinoctial elements the following process is used.

$$\begin{pmatrix} \Delta r = r_0 - r \\ \Delta v = v_0 - v \end{pmatrix} \cong \frac{\partial rv}{\partial Y} \Delta Y \qquad (10)$$

Where r_0 *and* v_0 are the initial position and velocity vectors, r *and* v are the SGP4 generated position and velocity vectors, $\frac{\partial rv}{\partial Y}$ is the partial derivative matrix of the position and velocity with respect to the equinoctial elements, and ΔY is the change in equinoctial elements. This equation provides the change in the position and velocity vectors based on the partial derivative of the position and velocity vectors with respect to the equinoctial element set multiplied by the change in the equinoctial element set. The partial derivative matrix is calculated by SGP4 and the change in position and velocity is generated with the first SGP4 generated vectors. The partials matrix is a 6x7 matrix containing the partial derivative of the position and velocity with respect to the equinoctial elements. This matrix is generated by calculating the numerical derivatives of the original orbit. One of the equinoctial elements is the drag coefficient, B^*, this element of the partials matrix is removed for the calculations presented in this study. The drag coefficient is not considered in this study since the objects are assumed to be point masses. By removing the drag coefficient from the partials matrix, the matrix is now a 6x6 square matrix.

The initial vectors, SGP4 generated vector and partials matrix are known. Solving for the change in equinoctial elements:

$$\Delta Y = \left(\frac{\partial rv}{\partial Y}\right)^{-1} \begin{pmatrix} \Delta r \\ \Delta v \end{pmatrix} \tag{11}$$

Once the change in equinoctial elements has been calculated, the previous set of equinoctial elements is incremented and run through the SGP4 process again until the position and velocity vectors are within 1×10^{-10} convergence. This process, called Newton-Raphson Iteration, provides the starting position (in equinoctial elements) of the target and victim satellites based on SGP4 calculations.

Propagation to the Maneuver Time

The next step is to propagate the target satellite to the maneuver time. The victim satellite does not require propagation; the movement of the target satellite will not affect the final position of the victim satellite. The maneuver time is given as some fixed increment prior to the close approach. The time can be given in minutes, hours, or fractions of periods. Any maneuver time can be provided. Multiple maneuver times are evaluated to see how the probability of collision changes based on the time the maneuver is performed. SGP4 has the set of equinoctial elements loaded into the scenario, it will use that set of elements and propagate the satellite back to a specified time prior to epoch. This action provides the position and velocity vectors at maneuver time and a partial derivative matrix of the position and velocity at maneuver time with respect to the equinoctial elements at the close approach time. This matrix is important for future calculations.

Deriving the Change in Position at the Close Approach Time

Once the target satellite has been propagated backwards, the change in the equinoctial elements at the close approach time can be calculated. Using the following equation

$$\begin{pmatrix} \delta r(t_m) \\ \delta v(t_m) \end{pmatrix} = \left(\frac{\partial \vec{r}, \vec{v}(t_m)}{\partial Y(t_c)} \right) \delta Y(t_c) \tag{12}$$

Where $\delta r(t_m)$ and $\delta v(t_m)$ are the change in the target satellite's position and velocity at the maneuver time, $\frac{\partial \vec{r}, \vec{v}(t_m)}{\partial Y(t_c)}$ is the partials matrix of the target position and velocity vectors at the maneuver time with respect to the equinoctial elements at the close approach time, and $\delta Y(t_c)$ is the change in the equinoctial elements at the close approach time due to the maneuver. This equation will give the change in the position and velocity vectors at the maneuver time. Since the change in the equinoctial elements at the close approach time is unknown, the change in the equinoctial elements at the conjunction time must be solved for in terms of the partials matrix and the change in the position and velocity at the maneuver time. At the maneuver time, the change in the position is zero since we are assuming the maneuver is impulsive; the change in velocity at the maneuver time is the desired maneuver magnitude. The change in the position at the maneuver time is zero because the location of the target satellite is determined by the propagation of the target satellite back to the maneuver time, this location will not change. Solving for the change in equinoctial elements at the close approach time

$$\delta Y(t_c) = \left(\frac{\partial \vec{r}, \vec{v}(t_m)}{\partial Y(t_c)}\right)^{-1} \begin{pmatrix} 0 \\ \delta v(t_m) \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} 0 \\ \delta v(t_m) \end{pmatrix} \qquad (13)$$

The partials matrix is broken into 4 different 3x3 matrices to make calculations simpler; **A**, **B**, **C**, and **D**. Therefore,

$$\delta Y(t_c) = \begin{pmatrix} B\delta v(t_m) \\ D\delta v(t_m) \end{pmatrix} \qquad (14)$$

This equation gives the change of the equinoctial elements at the close approach time with respect to the maneuver value at the maneuver time. Using the same method at the close approach time, the change in the position and velocity at the close approach time can be solved for since the change in the equinoctial elements has been calculated.

$$\begin{pmatrix} \delta r(t_c) \\ \delta v(t_c) \end{pmatrix} = \left(\frac{\partial \vec{r}, \vec{v}(t_c)}{\partial Y(t_c)}\right) \delta Y(t_c) \qquad (15)$$

Substituting in the value of $\delta Y(t_c)$:

$$\begin{pmatrix} \delta r(t_c) \\ \delta v(t_c) \end{pmatrix} = \left(\frac{\partial \vec{r}, \vec{v}(t_c)}{\partial Y(t_c)}\right) \begin{pmatrix} B\delta v(t_m) \\ D\delta v(t_m) \end{pmatrix} \qquad (16)$$

This equation solves for the change in the position and velocity at the close approach time with respect to the partials matrix at close approach time multiplied by the change in the equinoctial elements at the close approach time. The change in the velocity is not of interest for this study because the calculation of the probability of collision does not depend on the how the velocity changes at the close approach time due to the maneuver. Only the change in the position vector is needed to calculate the probability of collision for this method. The primary concern is how the maneuver changes the position at the close approach time. Breaking down the partials matrix again into four 3x3 matrices for ease of calculation; **E**, **F**, **G**, and **H**, one obtains the change in the position and velocity vectors at the close approach time with the equation below.

$$\begin{pmatrix} \delta r(t_c) \\ \delta v(t_c) \end{pmatrix} = \begin{pmatrix} E & F \\ G & H \end{pmatrix} \begin{pmatrix} B\delta v(t_m) \\ D\delta v(t_m) \end{pmatrix} = \begin{pmatrix} (EB + FD)\delta v(t_m) \\ not\ of\ concern \end{pmatrix} \tag{17}$$

The matrix $(\boldsymbol{EB} + \boldsymbol{FD})$ will be referred to as matrix J for the remainder of the calculations, simplifying the above equation into:

$$\delta r(t_c) = \boldsymbol{J}\delta v(t_m) \tag{18}$$

The above equation represents the change in the position of the target satellite at the close approach time based on the maneuver performed at the maneuver time.

Determining the Unconstrained Maneuver Direction

The probability of collision function given in *Modern Orbit Determination* by Wiesel is (Wiesel, 2010A):

$$P_{col} = \frac{1}{2\pi} |S_{rel}|^{-1/2} exp\left(-\frac{1}{2} \left(\Delta r + \delta r(t_c)\right)^T S_{rel}^{-1} \left(\Delta r + \delta r(t_c)\right) \right) \tag{19}$$

Where Δr is the distance between the victim and target satellites at the close approach time before the maneuver, miss distance, $\delta r(t_c)$ is the change in the position vector at the close approach time, and S_{rel} is the joint covariance matrix provided in the conjunction scenario. By performing a maneuver, the combined covariance matrix will be affected; however, for the purposes of this study it is assumed to remain constant to simplify calculations. To determine which direction to maneuver, the argument of the exponent is a quadratic function and can be maximized.

$$arg = \left(\Delta r + \boldsymbol{J}\delta v(t_m)\right)^T S_{rel}^{-1} \left(\Delta r + \boldsymbol{J}\delta v(t_m)\right) \tag{20}$$

Multiplying the equation out you get an equation for K:

$$K = \Delta r^T S_{rel}^{-1} \Delta r + \Delta r^T S_{rel}^{-1} \boldsymbol{J} \delta v(t_m) + \delta v(t_m)^T \boldsymbol{J}^T S_{rel}^{-1} \Delta r + \delta v(t_m)^T \boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J} \delta v(t_m) \tag{21}$$

K is the cost function that is maximized to determine which direction to maneuver. With this cost function, a constraint is added to ensure the target satellite will maneuver to a direction that will reduce the probability of collision. The constraint prevents the target from moving through the covariance matrix at any time, and only allows the target satellite to maneuver off to the side. The constraint implemented is:

$$\vec{v}_{rel} \cdot \left(\Delta r + \boldsymbol{J} \delta v(t_m)\right) = 0 \tag{22}$$

Where $\vec{v}_{rel}$ is the relative velocity between the target and the victim satellite at the close approach time. This method is referred to as the *unconstrained* maneuver direction because there are no constraints implemented to maintain any orbital parameters. The constraint added is necessary to avoid the fixed covariance matrix. This constraint forces the target satellite to perform and out-of-plane maneuver to reduce the probability of collision. The new cost function now adds the constraint:

$$K = \Delta r^T S_{rel}^{-1} \Delta r + \Delta r^T S_{rel}^{-1} \boldsymbol{J} \delta v(t_m) + \delta v(t_m)^T \boldsymbol{J}^T S_{rel}^{-1} \Delta r + \delta v(t_m)^T \boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J} \delta v(t_m) + \lambda \left(\vec{v}_{rel} \cdot \left(\Delta r + \boldsymbol{J} \delta v(t_m)\right)\right) \tag{23}$$

Where λ is the Lagrange multiplier. Taking the derivative of K with respect to the change in velocity at the maneuver time, the maximum of the equation is solved for:

$$\frac{\partial K}{\partial \delta v(t_m)} = (\Delta r^T S_{rel}^{-1} \boldsymbol{J})^T + \boldsymbol{J}^T S_{rel}^{-1} \Delta r + (\delta v(t_m)^T \boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J})^T + \boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J} \delta v(t_m) + \lambda(\vec{v}_{rel} \cdot \boldsymbol{J}) \tag{24}$$

Knowing that S_{rel} and S_{rel}^{-1} are symmetric matrices, the equation can be simplified to

$$\frac{\partial K}{\partial \delta v(t_m)} = 2\boldsymbol{J}^T S_{rel}^{-1} \Delta r + 2\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J} \delta v(t_m) + \lambda\left(\vec{v}_{rel} \cdot \boldsymbol{J}\right) \tag{25}$$

To solve for the maximum, set the derivative equal to zero and solve for $\delta v(t_m)$. This will give 3 linear equations, each one solving for the x, y, or z component of the change in velocity direction at the maneuver time. With the added constraint, the system has four equations and four unknowns.

$$\delta v(t_m) = -(\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J})^{-1} \boldsymbol{J}^T S_{rel}^{-1} \Delta r - \frac{1}{2}\lambda(\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J})^{-1}\left(\vec{v}_{rel} \cdot \boldsymbol{J}\right) \tag{26}$$

$$\begin{aligned} \begin{bmatrix} \delta v_x(t_m) \\ \delta v_y(t_m) \\ \delta v_z(t_m) \end{bmatrix} &= -([\boldsymbol{J}^T][S_{rel}^{-1}][\boldsymbol{J}])^{-1}[\boldsymbol{J}^T][S_{rel}^{-1}] \begin{bmatrix} \Delta r_x \\ \Delta r_y \\ \Delta r_z \end{bmatrix} \\ &\quad - \frac{1}{2}\lambda(\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J})^{-1} \begin{bmatrix} v_{relx} \\ v_{rely} \\ v_{relz} \end{bmatrix} \boldsymbol{J} \end{aligned} \tag{27}$$

The three equations above represent the first three equations in the system, the constraint, equation 22, makes the fourth equation for the system. With these four linear equations, each of the four unknowns is solved for and implemented to determine the direction to maneuver. The entire system of equations for the unconstrained maneuver direction is:

$$\begin{bmatrix} \\ -2\boldsymbol{J}^T S_{rel}^{-1} \Delta r \\ \\ -\vec{v}_{rel} \cdot \Delta r \end{bmatrix} = \begin{bmatrix} & v_x(t_m) \\ [2\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J}] & v_y(t_m) \\ & v_z(t_m) \\ [\vec{v}_{rel} \cdot \boldsymbol{J}] & 0 \end{bmatrix} \begin{bmatrix} \delta v_x \\ \delta v_y \\ \delta v_z \\ \lambda \end{bmatrix} \tag{28}$$

The cost function, when solved for $\delta v(t_m)$, yields the saddle point, a maximum. The equation solves for the direction of greatest increase, or the direction to the collision. If a maneuver is performed in this direction it will increase the chances the collision will occur. However, if the maneuver is performed in the opposite direction of the maximum, the maneuver magnitude can be chosen and maneuver in this direction to reduce the probability of collision. The components of $\delta v(t_m)$ are the optimal direction to maneuver. This direction will give the greatest increase in miss-distance with the least expenditure of fuel. This does not mean it is the only direction to maneuver to avoid the collision; it is the direction to maneuver to reduce the probability of collision the most efficiently.

Calculating the Constrained Maneuver Direction

The following equations describe the algorithm used in the constrained maneuver solution. The constrained maneuver applies the constraint to keep the change in the orbital energy equal to zero. The constraint equation is built as:

$$\Delta\varepsilon = 0 = \Delta\left(\frac{1}{2}v^2 - \frac{\mu}{r}\right) \tag{29}$$

$$= \left(\mu\frac{x}{r^3}, \mu\frac{y}{r^3}, \mu\frac{z}{r^3}, v_x, v_y, v_z\right)\begin{pmatrix} \delta r_x \\ \delta r_y \\ \delta r_z \\ \delta v_x \\ \delta v_y \\ \delta v_z \end{pmatrix} \tag{30}$$

Where ε is the orbital energy, μ is the gravitational parameter, $x, y, and\ z$ are the components of the position vector, $v_x, v_y, and\ v_z$ are the components of the velocity vector and r is the magnitude of the position vector. Since there is no change being implemented to the position of the target satellite at the maneuver time, the first three components of the 2nd vector are equal to zero. Simplifying the constraint to:

$$0 = \vec{v} \cdot \delta v \tag{31}$$

This orbital constraint is added to the unconstrained scenario to determine how the probability of collision will change while trying to maintain the same orbital period and semi-major axis. The addition of the second constraint to the cost function forces the satellite to perform a plane-change maneuver in order to maintain the current period and semi-major axis.

The constraint is added to the scenario and the cost function from equation 23. The new cost function with the constraint from equation 31 is:

$$\begin{aligned} K &= \Delta r^T S_{rel}^{-1} \Delta r + \Delta r^T S_{rel}^{-1} \boldsymbol{J} \delta v(t_m) + \delta v(t_m)^T \boldsymbol{J}^T S_{rel}^{-1} \Delta r \\ &+ \delta v(t_m)^T \boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J} \delta v(t_m) + \lambda_1 (\vec{v} \cdot \delta v) \\ &+ \lambda_2 \left(\vec{v}_{rel} \ \cdot \left(\Delta r + \boldsymbol{J} \delta v(t_m) \right) \right) \end{aligned} \quad (32)$$

Following the same process used to derive the unconstrained maneuver direction, the derivative of the cost function is taken to find the maximum of the equation.

$$\begin{aligned} \frac{\partial K}{\partial \delta v(t_m)} &= (\Delta r^T S_{rel}^{-1} \boldsymbol{J})^T + \boldsymbol{J}^T S_{rel}^{-1} \Delta r + (\delta v(t_m)^T \boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J})^T \\ &+ \boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J} \delta v(t_m) + \lambda_1 \vec{v} + \lambda_2 \left(\vec{v}_{rel} \ \cdot \boldsymbol{J} \right) \end{aligned} \quad (33)$$

Which simplifies to

$$\frac{\partial K}{\partial \delta v} = 2\boldsymbol{J}^T S_{rel}^{-1} \Delta r + 2\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J} \delta v(t_m) + \lambda_1 \vec{v} + \lambda_2 \left(\vec{v}_{rel} \ \cdot \boldsymbol{J} \right) \quad (34)$$

Where λ_1 and λ_2 are the Lagrange multipliers for the cost function. To solve for the maximum, set the partial derivative equal to zero and solve for $\delta v(t_m)$.

$$\delta v(t_m) = -(\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J})^{-1} (\boldsymbol{J}^T S_{rel}^{-1} \Delta r) - \frac{1}{2} \lambda_1 (\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J})^{-1} \vec{v}$$
$$- \frac{1}{2} (\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J})^{-1} \lambda_2 \left(\vec{v}_{rel} \cdot \boldsymbol{J} \right) \tag{35}$$

Rewritten in the components of the velocity and position components:

$$\delta v(t_m) = -(\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J})^{-1} (\boldsymbol{J}^T S_{rel}^{-1}) \begin{bmatrix} \Delta r_x \\ \Delta r_y \\ \Delta r_z \end{bmatrix} - \frac{1}{2} \lambda_1 (\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J})^{-1} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$
$$- \frac{1}{2} (\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J})^{-1} \lambda_2 \begin{bmatrix} v_{relx} \\ v_{rely} \\ v_{relz} \end{bmatrix} \boldsymbol{J} \tag{36}$$

The equation above represents the three linear equations for this scenario. There are five unknowns in the above equation. Adding the two constraint equations, equations 22 and 31, the system now has five linear equations and five unknowns. The full system of equations with constraints added is:

$$\begin{bmatrix} -2\boldsymbol{J}^T S_{rel}^{-1} \Delta r \\ \\ 0 \\ -\vec{v}_{rel} \cdot \Delta r \end{bmatrix} = \begin{bmatrix} \begin{array}{ccccc} & [2\boldsymbol{J}^T S_{rel}^{-1} \boldsymbol{J}] & & v_x(t_m) & \\ & (3x3) & & v_y(t_m) & \sum_\alpha \vec{v}_{rel_\alpha} \boldsymbol{J}_{\alpha i} \\ & & & v_z(t_m) & \\ v_x(t_m) & v_y(t_m) & v_z(t_m) & & 0 \\ & [\vec{v}_{rel} \cdot \boldsymbol{J}] & & & (2x2) \end{array} \end{bmatrix} \begin{bmatrix} \delta v_x \\ \delta v_y \\ \delta v_z \\ \lambda_1 \\ \lambda_2 \end{bmatrix} \tag{37}$$

When $\delta v(t_m)$ is solved for in this scenario, it will give the direction to the maxima of the system to maintain the same orbital energy. The same as the unconstrained maneuver direction, the opposite direction of the maxima is used to reduce the probability of collision. The components of $\delta v(t_m)$ are the optimal direction to maneuver. This direction will give the greatest increase in miss-distance with the least expenditure of fuel while maintaining the orbital period. This does not mean it is the only direction to maneuver to avoid the collision; it is the direction to maneuver to reduce the probability of collision the greatest while maintaining the constraint to keep the orbital period constant.

Probability of Collision for the Unconstrained and Constrained Maneuver

From equation 19, the probability or collision is calculated by

$$P_{col} = \frac{1}{2\pi} |S_{rel}|^{-1/2} exp\left(-\frac{1}{2}\big(\Delta r + \delta r(t_c)\big)^T S_{rel}^{-1}\big(\Delta r + \delta r(t_c)\big)\right) \tag{19}$$

The equation above is the probability of collision function for two objects in orbit. To determine the probability density for any given maneuver size, the direction calculated in the two previous sections, unconstrained and constrained maneuver directions, is normalized to remove the magnitude component.

$$\hat{n} = -\frac{\delta v_{col}}{|\delta v_{col}|} \tag{38}$$

Where δv_{col} is the maneuver direction. The opposite direction is taken to move away from the maximum. This normalized δv is multiplied times the δv expenditure, maneuver magnitude, resulting in the maneuver δv direction and magnitude. The maneuver δv is inputted into the probability of collision equation to determine the probability of collision for the time period and maneuver size. One of the assumptions of the methodology described above is the combined covariance matrix is treated as a constant for the calculations. The combined covariance matrix is fixed at the close approach time and the methodology provides a direction for the target satellite to maneuver around the covariance matrix, thus not crossing into the covariance matrix space.

Simulation Design

This research uses one collision scenario to perform all maneuver and probability calculations. The scenario used is a typical low Earth orbit (LEO) and is representative of satellites in LEO. Most potential collisions occur in the LEO range of orbits, the scenario selected allows this region of orbits to be evaluated. The hierarchy below shows the different solutions and what is presented in each.

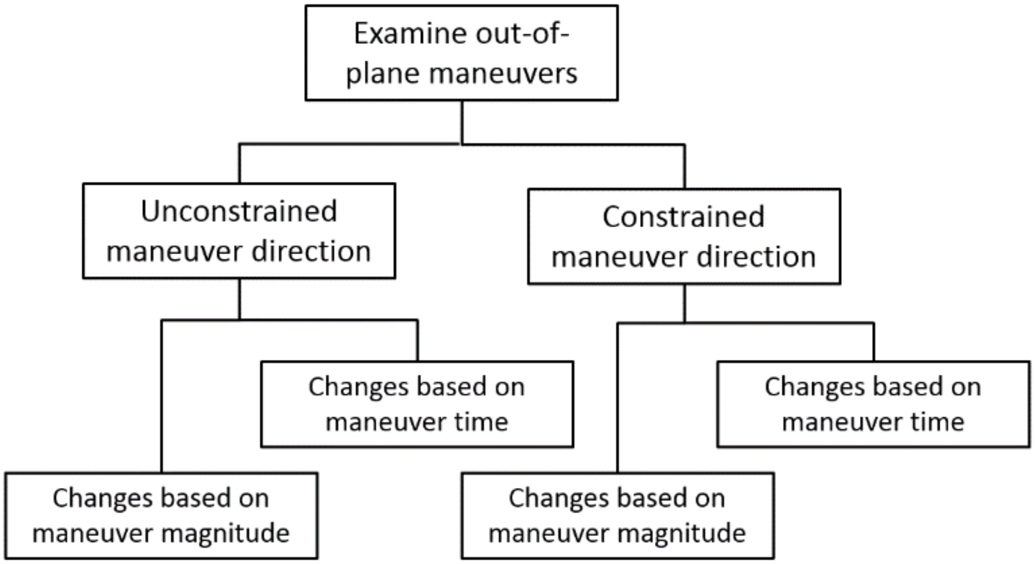

Examine out-of-plane maneuvers
Unconstrained maneuver direction
Constrained maneuver direction
Changes based on maneuver time
Changes based on maneuver magnitude
Changes based on maneuver time
Changes based on maneuver magnitude

Figure 5: Hierarchy of design methodology

The problem is constructed such that only out-of-plane maneuvers are considered. This is due to the constraint added to all analyses to ensure the target satellite does not pass through the fixed covariance matrix at any time. From the out-of-plane maneuvers, two different solution sets are presented, the constrained solution (where orbital period is maintained) and the unconstrained solution (where any out-of-plane maneuver is feasible). Each solution is then evaluated using two distinct sets of simulation parameters.

The first set evaluates changes based on maneuver magnitude, and the second evaluates changes based on maneuver time. Two different simulations are used because performing every possible combination of maneuver magnitude and time is not necessary for initial evaluation of maneuver magnitude and time analysis.

The first simulation, evaluates how changes in the magnitude of the collision avoidance maneuver affects the probability of collision. This simulation describes how the probability of collision changes for the scenario based on magnitude of maneuver in both the unconstrained and constrained directions. The described scenario performs different maneuver magnitudes at specific times preceding epoch based on the period of the target satellite. The following magnitudes of maneuver velocities were modeled:

Table 1: Maneuver Magnitudes Evaluated

0 m/s	1 cm/s	2 cm/s	3 cm/s	4 cm/s
5 cm/s	6 cm/s	7 cm/s	8 cm/s	9 cm/s
1 m/s	2 m/s	3 m/s	4 m/s	5 m/s
6 m/s	7 m/s	8 m/s	9 m/s	10 m/s

Each maneuver magnitude is evaluated at different time periods. This shows how a different magnitude of maneuver changes the probability of collision based on the maneuver performed at different times. Table 2 below identifies the time periods modeled. The maneuver times selected represent various different fractions of orbits and whole orbits back to five times the period preceding epoch.

Table 2: Period Fractions Evaluated

1/4 period	1/3 period
1/2 period	2/3 period
3/4 period	1 period
1 1/4 period	1 1/3 period
1 1/2 period	2 periods
2.5 periods	3 periods
4 periods	5 periods

The maneuver magnitudes selected represent a wide range of differing maneuver magnitudes for a collision avoidance maneuver. These values are selected to give a broad range of how changing the maneuver magnitude at different times in the orbit will affect the probability of collision. The list is not exhaustive, but provides enough information to show how different time periods respond to different maneuver magnitudes. At each time step shown above, every maneuver magnitude listed in Table 1 is evaluated to determine how the magnitude of the maneuver changes the probability of collision at that time. The process is then repeated for each time step, for both the constrained and unconstrained maneuver directions.

The second simulation focuses on how the target satellite’s probability of collision changes based on performing a fixed maneuver magnitude each second prior to epoch. This simulation performs the probability of collision calculation at each second preceding the epoch time for a fixed maneuver of 1 meter per second. The simulation was performed at the time step of one second because the difference in the probability density between fractions of seconds is negligible. The maneuver magnitude of 1 meter per second was selected as a reasonable representation of what a spacecraft operator is willing to expend for a collision avoidance maneuver. The results from the second simulation show what times to maneuver based on the probability of collision calculation. This is different than the first method since it looks at a finer time scale for the target satellite preceding epoch. This process is performed for both the constrained and unconstrained maneuver directions.

Summary

This chapter provides the background information on the methodology used for this thesis. The chapter describes how the scenario was loaded into the SGP4 propagator. The method used to propagate the target satellite to the maneuver time is presented. The algorithms used for determining which direction to maneuver the target satellite was discussed and calculating the probability of collision given the maneuver direction and magnitude was presented. Finally, the simulation design describing how each solution is evaluated is presented.

IV. Analysis and Results

Chapter Overview

This chapter will provide the results that follow the methodology presented in the previous chapter. The first set of results will show how the probability of collision changes with the magnitude of the maneuver chosen. The final set of results reflect how the probability of collision changes when using the same maneuver but varying the maneuver time within the orbital period. All code used to support the results presented below is attached in Appendix A.

Scenario Set Up

The conjunction scenario used for this thesis is generated to ensure a high likelihood the target and victim satellites collide. The scenario parameters are

- Target Position: $\langle 7078.14, 0, 0\rangle$ kilometers in inertial frame
- Target Velocity: $\langle 0.2, 7.5, 0.2\rangle$ kilometers per second in inertial frame
- Victim Position: $\langle 7078.15, 0, 0\rangle$ kilometers in inertial frame
- Victim Velocity: $\langle 0.2, 7.5, 0.2\rangle$ kilometers per second in inertial frame
- Close Approach Time (YYYY:MM:DD:HH:MM:SS.SS): 2013:11:01:22:45:02.43
- Joint Covariance Matrix: $\begin{bmatrix} 0.005 & 0 & 0 \\ 0 & 0.005 & 0 \\ 0 & 0 & 0.005 \end{bmatrix} kilometers^2$

These parameters are loaded into the conjunction file that the program reads. This is the only external information about the close approach required for the simulation and all of the above information is available on a CSM. The figure below depicts the collision scenario described above, figure not to scale.

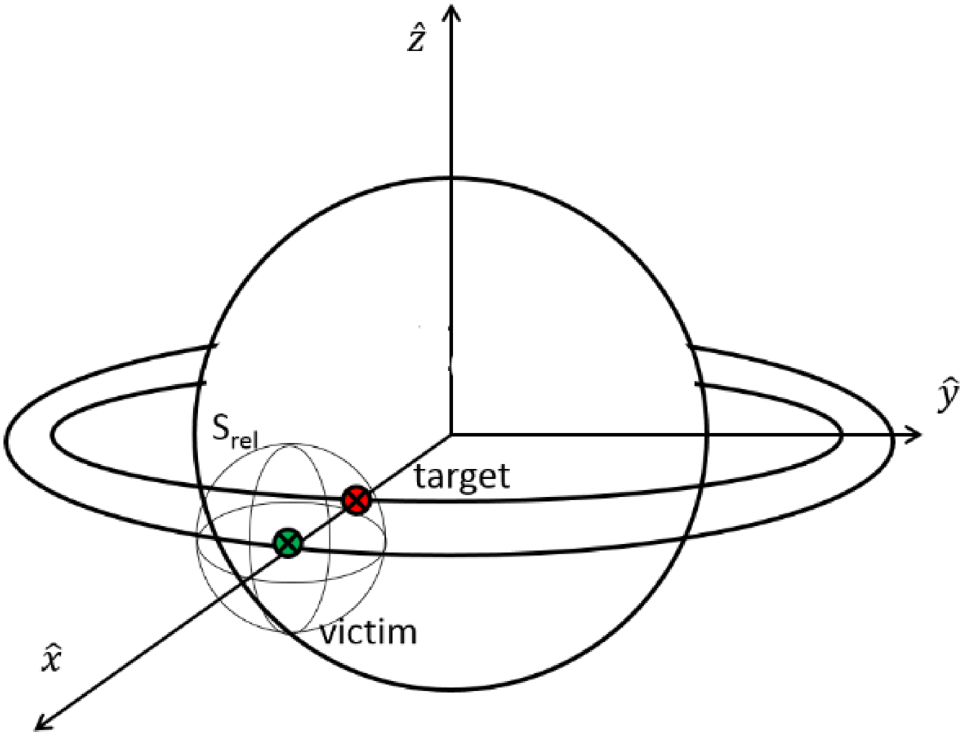


Figure 6: Collision Scenario

These parameters were selected to create a close approach situation where the target and victim satellites are separated by only 10 meters. The target satellite is in an orbit 10 meters closer to the Earth than the victim satellite. The joint covariance matrix was created based on analyzing actual CSMs to get a good approximation for covariance matrices that are currently in use. The covariance matrix is a 3-dimensional Gaussian. The probability distribution has infinitely long tails, so the boundary of the spherical structure is set to 10×10^{-9}, this study assumes this to be an acceptably low probability. The miss-distance with the covariance matrix created a situation where the uncertainty in the satellite's position exceeded the actual difference they are apart; this generated a case where the two satellites would most likely collide at the close approach time. The following classical orbital elements are generated from the input of the initial position and velocity vectors for the target and victim satellites.

Table 3: Target and Victim initial Classical Orbital Elements

Classical Orbital Element	Target Satellite	Victim Satellite
Semi-Major Axis (a)	7080.11 km	7080.13 km
Period (P)	5928.86 s	5928.89 s
Mean Motion (n)	0.0635857 rad/s	0.0635854 rad/s
Mean Anomaly (M)	1.53368 = 87.87°	1.53363 = 87.87°
Eccentricity (e)	0.02665	0.02665
Argument of Perigee (ω)	4.6961 = 269.07°	4.6962 = 269.07°
Inclination (i)	0.0266 = 1.52°	0.0266 = 1.52°
Right-Ascension of the Ascending Node (Ω)	0.00 = 0.00°	0.00 = 0.00°

Changes based on Maneuver Magnitude

Both the unconstrained maneuver and the constrained maneuver solutions are investigated for changes based on maneuver magnitude. For the unconstrained maneuver, the probability of collision graph over discrete time periods for varying magnitudes of maneuver is below.

Figure 7 shows that, for this scenario, there are different times during the orbit of the target satellite that will provide the greatest reduction in probability density for the smallest fuel expenditure.

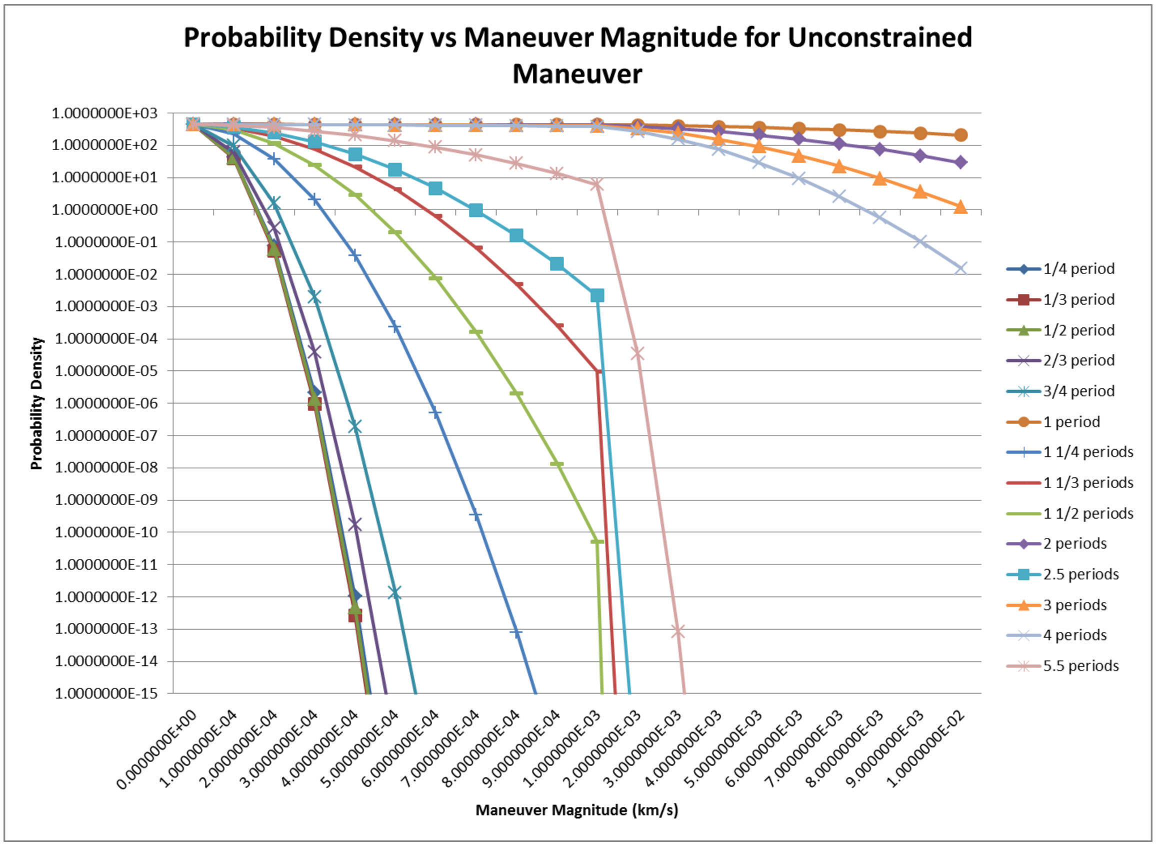


Figure 7: Probability Density for Unconstrained Maneuver Direction

For the times of 1/4, 1/3, and 1/2 period, a 4 centimeter per second maneuver magnitude is required at those time periods to reduce the probability density to less than 1×10^{-9}. The longer the maneuver precedes epoch the greater the fuel expenditure required to reach the minimum probability density. This is due to the scenario presented is in LEO and perturbing forces, like drag, affecting the orbit. The more the maneuver precedes epoch, the longer these perturbing forces act on the orbit. If a small maneuver is performed days prior to epoch, the perturbing forces will cause the orbit to move back closer to the original orbit. Due to this effect, the results show that if the small maneuver is performed long before epoch it does not cause a large change in the probability density. Instead, a larger maneuver must be performed to reduce the probability density. The plots across the top show the probability of collision if the maneuver were to be performed at 1, 2, 3 and 4 periods prior to the close approach. No matter how large a maneuver at these period points, the probability of collision is not reduced to a negligible level, it is still over 1×10^{-9}. The position of the target satellite will remain in approximately the same location no matter what magnitude of velocity is performed during these incremental time periods. This is the reason that performing the maneuver at one period prior to the close approach, even a larger maneuver (10 meters per second), will not change the probability that the two satellites will collide. Since the constraint added to the scenario forces the satellite to perform an out-of-plane maneuver, incremental period times are ineffective to reducing the probability of collision. This constraint eliminates the option to perform a phasing maneuver so that the target satellite does not fly through the covariance matrix at any time, instead around the matrix at all times.

For the constrained maneuver--the situation where the direction of the maneuver is dictated to maintain the current energy of the orbit, thus not changing the orbital period--the following graph shows how the probability of collision changes for different magnitudes and different time periods.

Figure 8 for the constrained scenario shows more fuel expenditure is required to attain the same probability density of 1×10^{-9} than the unconstrained scenario.

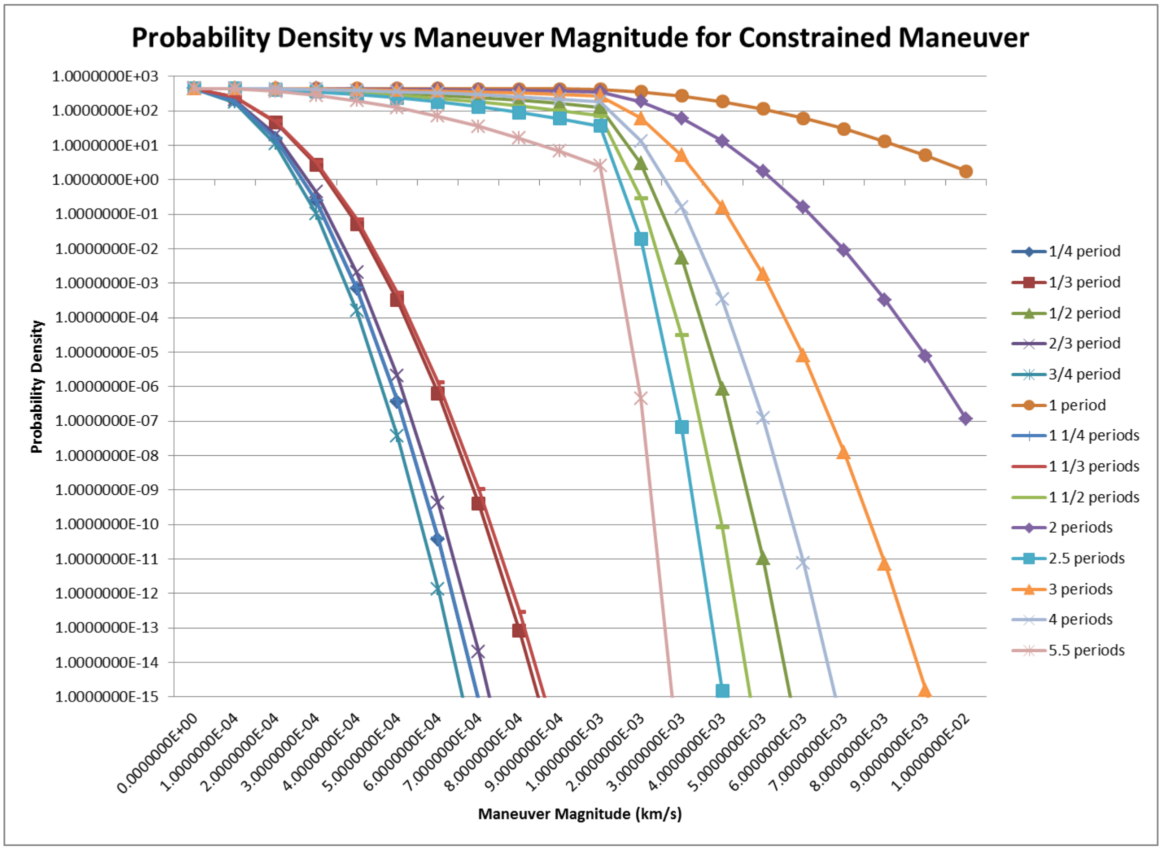


Figure 8: Probability Density for Constrained Maneuver Direction

For 1/4 period, 2/3 period, 3/4 period, and 1 1/4 periods the fuel expenditure required is 6 centimeters per second to reduce the probability density to 1×10^{-9}. For 1/3 period and 1 1/3 periods the minimum fuel expenditure required is 7 centimeters per second. These results are clustered together on the graph and reduce the probability density the greatest for the least amount of fuel for the constrained scenario. The next four results to the right on the graph represent the time periods in increments of half periods; 5 1/2 periods, 2 1/2 periods, 1 1/2 periods, and 1/2 period. With these four time periods, the probability density does reduce to acceptable levels, but it requires more than 3 meters per second fuel expenditure. For these four time periods, the further away from the close approach time the less fuel that is required to reduce the probability density to 1×10^{-9}.

These time periods would not be effective if the system only simulated the two body problem. Increments of the half period produce results of reducing probability because of nodal regression perturbing the orbit. This is why increase in lead time the maneuver is performed (5 1/2 periods) it requires the least amount of fuel for these four time periods. The next four results to the right in Figure 8 represent the maneuver performed at incremental periods; 4 periods, 3 periods, 2 periods, and 1 period. Similar to the half period points, these time periods require less fuel expenditure the longer the maneuver is performed preceding epoch. For the maneuver magnitudes investigated in this study, only the 3 and 4 period times showed a reduction in probability density to 1×10^{-9}. The 4 period time requires a 6 meter per second maneuver magnitude and the 3 period time requires an 8 meter per second maneuver magnitude. For both the 2 period and 1 period times, much greater than a 10 meter per second maneuver magnitude is required, which is beyond what is typical for satellite collision avoidance maneuvers. The half period and period points are not as effective in reducing the probability density because the maneuver performed in the constrained scenario is a plane change maneuver. The maneuver is perpendicular to the position and velocity vectors; this is how the orbital period is maintained. When this type of maneuver is performed at the half period and period points in the two body problem, the location of the half period and period points do not change. In the scenario here, other perturbing effects, such as nodal regression, allow the half period and period points to move, thus making these time periods a valid way to reduce collision probability.

Changes based on Maneuver Time

Both the constrained and unconstrained maneuver direction solutions are investigated with the changes based on maneuver time. The figure below shows the results of the unconstrained maneuver direction evaluated under the changes based on maneuver time method.

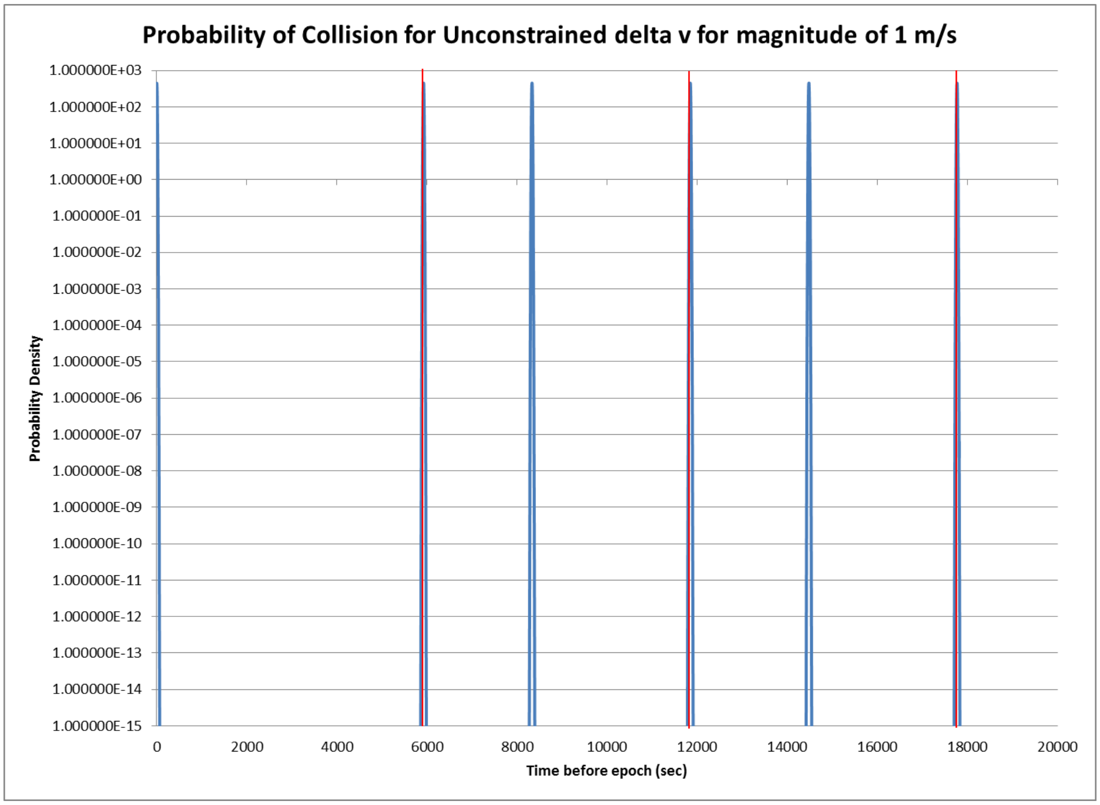


Figure 9: Probability over Time for 3 Periods Unconstrained Maneuver Direction

Figure 9 above depicts the probability density calculations for three periods prior to epoch for a 1 meter per second maneuver magnitude occurring at each second. The red lines delineate the incremental period points. For this scenario the period of the orbit is 5928 seconds, approximately 99 minutes. The graph shows the probability of collision for each second prior to the epoch time. The graph is cut off below 1×10^{-15}, because a probability of collision less than 1×10^{-15} is essentially zero. From Figure 9 above, one can see that there are non-optimal times to maneuver. Where the graph peaks are the non-optimal times to maneuver. These times correspond with the incremental period points and prior to the half period mark. The peaks of the graph are shown in the table below. These times are the non-optimal times of maneuver. There is a non-optimal time window to maneuver of 100 seconds to maintain a probability of collision of less than 1×10^{-9} with these values at the middle.

Table 4: Non-optimal maneuver times for unconstrained maneuver direction

Time (seconds)	Point in period
5922	6 seconds prior to 1 period before epoch
8330	562 seconds prior to 1.5 periods before epoch
11843	13 seconds prior 2 periods before epoch
14475	345 seconds prior to 2.5 periods before epoch
17762	22 seconds prior to 3 periods before epoch

The longer preceding epoch the maneuver occurs, the spike between orbits of non-optimal maneuver moves closer to the half period point and maintains a time of non-optimal maneuver at prior to half a period. If the scenario is run for a longer simulation time, the time of non-optimal maneuver at the half period point moves to the actual half period point. The model takes into account orbital perturbations; the closer the target satellite is to the epoch time the less the perturbations will affect the maneuver performed. Figure 9 shows that if you maneuver in the unconstrained direction at any time other than the period and prior to the half period points by 1 meter per second you reduce your probability of collision to be a negligible amount. The half period points are similar to the period points where maneuvering does not provide any noticeable benefit to reducing the probability because the maneuver at the half period point is not having an effect on the collision position. The amount of the maneuver is not changing the orbital parameters enough to make a difference at the collision location. One thing to note about the unconstrained maneuver direction is that at the half period point prior to the epoch time, there is not a window of non-optimal maneuver. This shows that even though the half period point is a non-optimal time that the potential for collision is close enough to where the perturbing effects of the orbit are not going to negate the changes made by the maneuver.

Figure 10 below shows the same situation except the direction of the maneuver is constrained to maintain the orbital period after the maneuver. The maneuver magnitude is still a 1 meter per second maneuver magnitude at each second for three periods prior to epoch. Figure 10 shows how the probability of collision changes when the maneuver direction is perpendicular to the velocity direction maintaining the same orbital period. The windows of non-optimal maneuver times are the same size as the unconstrained maneuver direction.

The table below, Table 5, shows the peaks of non-optimal maneuver time for the constrained maneuver direction. For the constrained maneuver direction, each of these time points has a 100 second window where the probability of collision is greater than 1×10^{-9}. The half period mark for the constrained situation is closer to the half period. This is due to the direction of maneuver represented here is a plane-change maneuver.

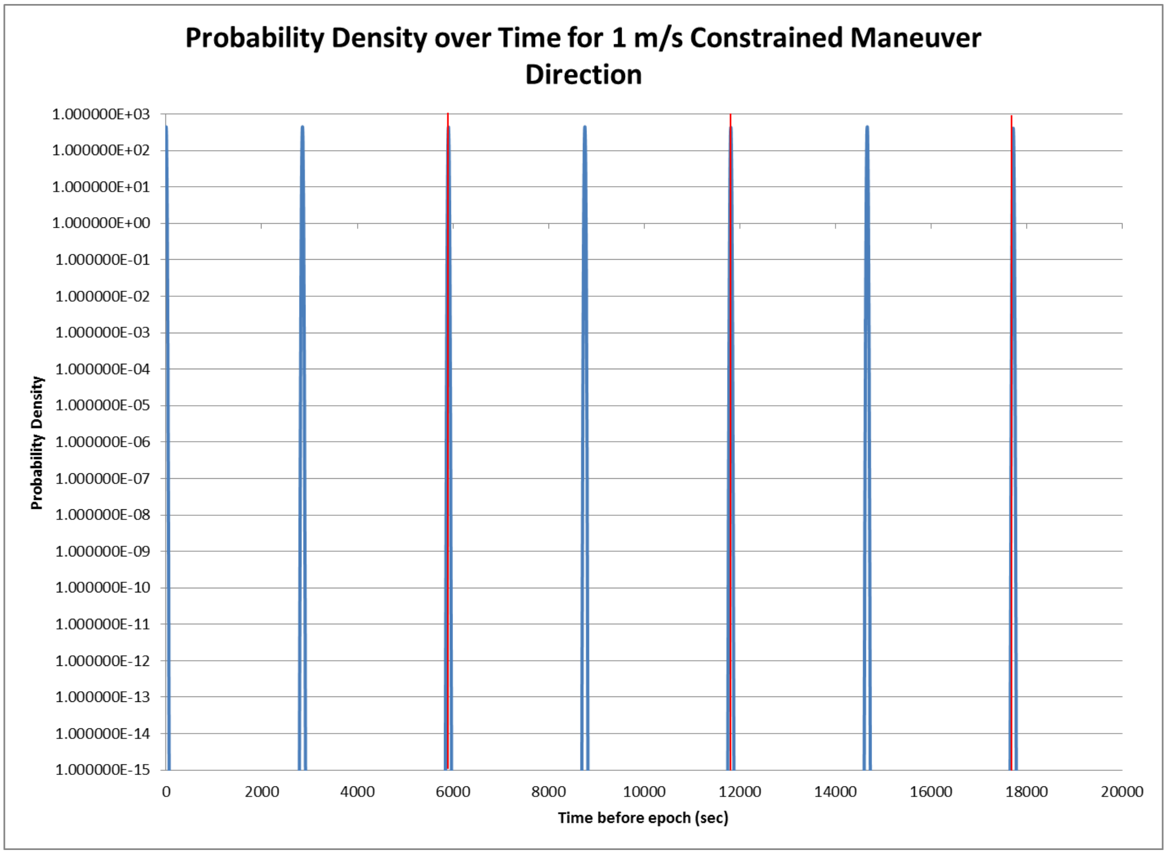


Figure 10: Probability over Time for 3 Periods Constrained Maneuver Direction

Table 5: Non-optimal maneuver times for constrained maneuver direction

Time (seconds)	**Point in period**
2853	111 seconds prior to 0.5 periods before epoch
5904	24 seconds prior to 1 period before epoch
8758	134 seconds prior 1.5 periods before epoch
11810	46 seconds prior to 2 periods before epoch
14663	157 seconds prior to 2.5 periods before epoch
17715	39 seconds prior to 3 periods before epoch

The plane-change will change the orientation of the orbit. The line that the orbit pivots on for the maneuver is the radial vector through the center of the earth. For this reason, if the maneuver occurs at the half period prior to epoch the location of the collision will not change. The line that the orbit is pivoting on will go through the center of the earth and align with the period point, not making a change to either of those locations. Since the constrained maneuver direction represents a plane-change maneuver, maintaining the orbital period and semi-major axis, there is time of non-optimal maneuver at the half period point prior to epoch. This is different from the unconstrained scenario; it is different since the direction of the constrained maneuver is perpendicular to the orbital plane.

For this simulation, a fixed maneuver magnitude is used to calculate the probability of collision at each second prior to epoch; but these results may be extrapolated to determine the effect of different magnitudes as discussed below. In both of the solutions listed above, if the magnitude of the maneuver is increased, the windows of non-optimal maneuver times would shrink. If the maneuver magnitude is decreased, the windows of non-optimal maneuver time would increase and encompass the period and half period points more. The same patterns would emerge that at the period and half period points the probability of collision would be non-optimal, but the amount of non-optimal time to maneuver would change depending on an increase or a decrease of maneuver magnitude. The values listed above are directly tied to the scenario created for this study. The location of the non-optimal maneuver times would remain at the period and half period points, but the non-optimal window and timing would be different based on the collision scenario.

Summary

This chapter described the results of the different simulations. The first simulation investigated is the unconstrained and constrained maneuver directions and how varying magnitudes of maneuvers changed the probability of collision at discrete points in a satellite's orbital period. The next simulation was run with a fixed maneuver magnitude being performed at all seconds prior to the epoch time for the unconstrained and constrained solutions.

V. Conclusions and Recommendations

Chapter Overview

This chapter presents the conclusions and recommendations for the research. Each research goal is reviewed, demonstrating that they have been met. The significance of the research and how it relates to the space community is discussed. The recommendations for future work are presented.

Conclusions of Research

The focus of this thesis was to determine how a satellite's probability of collision will change by varying the maneuver magnitude and direction. The study focused on three research questions which are answered below.

1. **Estimate the probability of collision changes from different magnitudes of maneuver.**

The analysis from chapter 4 shows that for both the constrained and the unconstrained maneuver directions, the probability of collision can be reduced to acceptable amounts with a reasonably small maneuver. For the unconstrained scenario the minimum maneuver magnitude required is 4 centimeters per second to reduce the probability density to 1×10^{-9}. For the constrained scenario, maintaining the orbital period, the minimum maneuver magnitude required is 6 centimeters per second. Different time periods in both scenarios respond differently to the maneuver magnitudes. In the unconstrained scenario, the worst times to maneuver occur at incremental periods. Each of the investigated incremental time periods showed that the probability density does not reduce to acceptable limits even with a 10 meter per second maneuver. In the constrained scenario, the worst times to maneuver occur at incremental period and half periods. If a large enough maneuver is performed, the probability can come within acceptable limits, however, there are more fuel efficient times to maneuver. What is most interesting is that in both the constrained and unconstrained scenarios, there are options to maneuver within one period of the close approach time that do not require large expenditures of fuel. This means, even with little notice, collision avoidance maneuvers can be performed up to minutes prior to the collision time to reduce the probability to almost negligible amounts.

2. Determine optimal maneuver direction and maneuver times to reduce probability of collision

This study determined that, for a 1 meter per second fuel expenditure, if the target satellite is to maneuver at *N* periods prior to the potential collision then the probability that the two objects will collide remains unchanged. It is also determined that if the maneuver occurs around the half period time the probability of collision will remain unchanged. Both of these time periods are non-optimal times to maneuver and will not gain the target satellite a reduction in probability of collision. What is interesting is that if the maneuver is performed at any other time during the orbit the probability density is reduced to negligible amounts. This same trait occurred in both the unconstrained and constrained situations. Since the probability equation had only one maximum, there is no *optimal* time to maneuver, there is no minimum probability the satellite is trying to reach. This is why the information is presented as finding the non-optimal maneuver time and magnitudes.

3. Determine optimal maneuver direction to reduce the impact on certain orbital parameters while reducing probability of collision

The constrained maneuver direction situation has shown the best direction to maneuver to reduce the impact on orbital period. The analysis shows that if the maneuver for the constrained direction is performed at any time other than the period or half period, the orbital period of the target satellite remains unchanged while reducing the probability density. If the maneuver is performed in one of these locations, the orbital period will remain the same but it will not affect the probability of collision unless a large maneuver magnitude is used, greater than 5 meters per second. This simulation has also shown that for the constrained maneuver direction, a maneuver magnitude of 6 centimeters per second can make a drastic difference in the probability density, if the maneuver is performed at any time other than the period and half period times.

Significance of Research

This research is significant since there is currently no tool to provide operators with an optimal direction, time and magnitude, to perform a collision avoidance maneuver. The methodology evaluates what direction the target satellite should maneuver in order to significantly reduce the probability of collision and potential impacts on orbital parameters. Only one orbital parameter was evaluated in this research, however, it is shown there is a way to maneuver and maintain the orbital period and reduce the probability density. The other significant aspect of this research is it provides a way to look at how varying magnitudes of a maneuver can change the probability density. Depending on what is more important to the satellite owner. Having both tools to determine how best to maneuver to maintain a constellation is very pertinent to satellite operators. A review of the salient literature reveals that no method exists reducing the probability of collision by performing a collision avoidance maneuver. A lot of the research has shown how to calculate the probability of collision based on a given situation and how to provide a better estimate of a covariance matrix to make the probability of collision estimates more accurate. This research goes one step further to give information to the satellite operator about how to maneuver a satellite to reduce the probability of collision.

Recommendations for Future Research

Only one collision scenario was evaluated at varying magnitudes and maneuver times and only one orbital parameter constraint was applied. Future research should include running the simulation for different collision scenarios to show its effectiveness in other orbits. The future work should also include applying more constraints to the methodology to limit the effect on other orbital parameters by performing a maneuver. Future research should include the uncertainty introduced from the maneuver itself. This will provide a better estimate and more accurate answer for the probability of collision at the close approach time. The assumption made that the covariance matrix is fixed at the close approach time should be discarded in future research. This assumption was made to generate a direction for the target satellite to maneuver around the covariance structure, when in actuality, the covariance matrix will move based on which direction the target satellite maneuvers. Investigating how to include these effects will enhance the efficacy of the methodology, possibly identifying even more fuel efficient maneuver parameters.

Summary

The environment in which satellites operate is constantly changing and the amount of space that each satellite has to operate in is decreasing as more objects are launched. Collisions on orbit do not occur frequently, but many satellites are threatened with close approaches. This thesis presents a methodology to give satellite operators additional information about their options when in a close approach scenario. The methodology provided in this thesis provides information about which direction is optimal to maneuver and how much fuel needs to be expended to provide the satellite operator with the ability to eliminate the probability of collision. The information provided in this thesis is going to become more pertinent to satellite operators as more objects are launched into space and the environment becomes more congested.

Appendix A – Code written to support research

```
// TestMatrix.cpp : Defines the entry point for the console application.
/* TestMatrix performs the probability of collision calculation for the
unconstrained and constrained
maneuver directions at discrete points in the period for varying velocity
magnitudes*/

#include "stdafx.h"
#include <iostream>
#include <stdio.h>
#include "Sgp4.h"
#include "sgp4unit.h"
#include "sgp4io.h"
#include "sgp4ext.h"
#include "ludcmp.h"
#include "JulianDay.h"


int _tmain()
{

double r0target[3], rtarget[3], rtargetb[3];
double v0target[3], vtarget[3], vtargetb[3];
double r0victim[3], rvictim[3];
double v0victim[3], vvictim[3];
double p0target, p0victim,ptargetm;
double a0target, a0victim, atargetm;
double ecc0target, ecc0victim, ecctargetm;
double incl0target, incl0victim, incltargetm;
double omega0target, omega0victim, omegatargetm;
double argp0target, argp0victim,argptargetm;
double nu0target, nu0victim, nutargetm;
double m0target, m0victim, mtargetm;
double arglat0target, arglat0victim, arglattargetm;
double truelon0target, truelon0victim, truelontargetm;
double lonper0target, lonper0victim, lonpertargetm;
double P0target, P0victim, Ptargetm;
double X0target[7], Y0target[7], X0targetS[7], XtargetS[7], YtargetS[7],
YtargetSn[7], Ytarget[7];
double X0victim[7], Y0victim[7], X0victimS[7], XvictimS[7], YvictimS[7],
YvictimSn[7], Yvictim[7];
double n0target, n0victim, ntargetm;
double bstart, bstarv;
class Spg4;
int satnot, satnov;
double epocht, Y0[7], Yst[7], Ysv[7], Ytargetb[7], Xtargetb[7];
double drvdYtb[6][7];//partials matrix at tbackepoch wrt EOE at tclose
double* drvdYtbptr = drvdYtb[0];
double drvdYtm[6][7];//partials matrix at tmaneuver wrt EOE at tclose
double* drvdYtmptr = drvdYtm[0];
double drvdYtc[6][7];//partials matrix at tclose wrt EOE at tclose
double* drvdYtcptr = drvdYtc[0];
double timet; //how far do we want to propogate the orbits
double epochtn; //new epoch time for taget
int year, month, day, hour, minute;
```

```
double second;
double deltart[3], deltavt[3], deltarvt[6], deltaYt[6];
double deltarv[3], deltavv[3], deltarvv[6], deltaYv[6];
double row[6];
MatDoub Srel(3,3), Srelinverse(3,3), JTSreliJ(3,3), invJTSreliJ(3,3), matJ(3,3),
invJ(3,3);//joint covariance matrix for target and victim
double matB[3][3], matD[3][3], matE[3][3], matF[3][3], matG[3][3], matH[3][3],
matEB[3][3], matFD[3][3];
double matJT[3][3], SreliJ[3][3], JTSreli[3][3], iJTSiJJTSi[3][3];
double missr[3], delv[3], delvc[3];
double Jdelv[3], Jdelvc[3];
double delrJdelv[3], delrJdelvc[3];
double SrelidelrJdelv[3], SrelidelrJdelvc[3], iJTSJvtb[3];
double rJvSrJv, lambda, rJvSrJvc;
double magSrel, magSrelc;
double coeff, coeffc;
double Pcol[20], Pcolc[20];
double vrel[3], iJTSJJvrel[3], vrelJ[3];

       FILE* Probability;
       Probability = fopen("Probability.txt", "w");

       FILE* Constraint;
       Constraint = fopen("Constraint.txt", "w");

    FILE* pInput;
    pInput = fopen( "Conjunction1.txt", "r");
    fscanf( pInput, "%i %i %Le %Le %Le %Le %Le %Le %Le %Le %Le %Le %Le %Le %Le %Le
%i %i %i %i %i %Le %Le %Le %Le %Le %Le %Le %Le %Le %Le",
            &satnot, &satnov, &bstart, &bstarv, &r0target[0], &r0target[1],
&r0target[2], &v0target[0],
            &v0target[1], &v0target[2],
            &r0victim[0], &r0victim[1], &r0victim[2], &v0victim[0], &v0victim[1],
&v0victim[2], &year, &month,
            &day, &hour, &minute, &second, &Srel[0][0], &Srel[0][1], &Srel[0][2],
&Srel[1][0], &Srel[1][1],
                     &Srel[1][2], &Srel[2][0], &Srel[2][1], &Srel[2][2] );

    //rv2coe from Sgp4ext require mu for the gravitational parameter-mu of earth
398600.5 km^3/s^2
    double mu = 398600.5;

    //convert YMDHMS to Julian Day
    JulianDayNumber(year, month, day, hour, minute, second, epocht);

    //Have to run everything from here down for the target satellite and the
victim satellite
       //rv2coe should give back classical elements from a given r and v

       //target satellite
    rv2coe(r0target, v0target, mu, p0target, a0target, ecc0target, incl0target,
omega0target,
           argp0target, nu0target, m0target, arglat0target, truelon0target,
lonper0target);
       //victim satellite
       rv2coe(r0victim, v0victim, mu, p0victim, a0victim, ecc0victim, incl0victim, 
omega0victim,
           argp0victim, nu0victim, m0victim, arglat0victim, truelon0victim, 
lonper0victim);
       //need to conver r and v from EFG to IJK to use rv2coe
    //once you have converted your r0 and v0 to classical elements, create the 7x1 
classical vector (X) containing air drag term (input from conjunction message)
       //target
    P0target = (2.0 * pi) * pow((pow(a0target, 3.0) / mu), 0.5); //calculate 
initial period from calculated COEs
    n0target = (pow(mu/pow(a0target,3.0),0.5))*60; //calculate mean motion for 
your satellite
    X0target[0] = n0target;
    X0target[1] = m0target;
    X0target[2] = bstart;
    X0target[3] = ecc0target;
    X0target[4] = argp0target;
    X0target[5] = incl0target;
    X0target[6] = omega0target;
       //victim
       P0victim = (2.0 * pi) * pow((pow(a0victim, 3.0) / mu), 0.5); //calculate 
initial period from calculated COEs
    n0victim = (pow(mu/pow(a0victim,3.0),0.5))*60; //calculate mean motion for 
your satellite
    X0victim[0] = n0victim;
    X0victim[1] = m0victim;
    X0victim[2] = bstarv;
    X0victim[3] = ecc0victim;
    X0victim[4] = argp0victim;
    X0victim[5] = incl0victim;
    X0victim[6] = omega0victim;

    //now need to run SGP4 on the above X vector, this will create the Y vector of 
equinoctal elements and the drvdy matrix needed
    //Run SGP4 at the same time for it to give you the estimated equinoctal 
elements for the given r0 and v0

    FILE* pDebug;
       pDebug = fopen("DebugFile.txt", "w");
       fprintf( pDebug, " rOTarget %21.14Le  %21.14Le  %21.14Le\n", r0target[0], 
r0target[1], r0target[2] );
       fprintf( pDebug, " vOTarget %21.14Le  %21.14Le  %21.14Le\n\n", v0target[0], 
v0target[1], v0target[2] );
       fprintf( pDebug, " a0 target %21.14Le P0target  %21.14Le\n", a0target, 
P0target );
       fprintf( pDebug, " XOTarget %21.14Le  %21.14Le  %21.14Le\n", X0target[0], 
X0target[1], X0target[2] );
       fprintf( pDebug, " XOTarget %21.14Le  %21.14Le  %21.14Le\n", X0target[3], 
X0target[4], X0target[5] );
       fprintf( pDebug, " XOTarget %21.14Le\n", X0target[6] );
       fprintf( pDebug, " tepoch %21.14Le \n\n", epocht );

       fprintf( pDebug, " rOvictim %21.14Le  %21.14Le  %21.14Le\n", r0victim[0], 
r0victim[1], r0victim[2] );
       fprintf( pDebug, " vOvictim %21.14Le  %21.14Le  %21.14Le\n\n", v0victim[0], 
v0victim[1], v0victim[2] );
```

```
       fprintf( pDebug, " a0 victim %21.14Le P0victim  %21.14Le\n", a0victim,
P0victim );
       fprintf( pDebug, " XOvictim %21.14Le  %21.14Le  %21.14Le\n", X0victim[0],
X0victim[1], X0victim[2] );
       fprintf( pDebug, " XOvictim %21.14Le  %21.14Le  %21.14Le\n", X0victim[3],
X0victim[4], X0victim[5] );
       fprintf( pDebug, " XOvictim %21.14Le\n", X0victim[6] );
       fprintf( pDebug, " tepoch %21.14Le \n\n", epocht );
       //if( 1 != 0 ) exit(0);

       //run iterate for target satellite
    SimpGenPert4 perttarget;   // fix to one class variable
       perttarget.XtoY(X0target, Y0target);

       fprintf( pDebug, " Target Satellite initial equinoctal state" );
       fprintf( pDebug, "\n\n Xo -> Y0:  %21.14Le  %21.14Le  %21.14Le\n",
Y0target[0], Y0target[1], Y0target[2] );
    fprintf( pDebug, " Xo -> Y0:  %21.14Le  %21.14Le  %21.14Le\n", Y0target[3],
Y0target[4], Y0target[5] );
       fprintf( pDebug, " Xo -> Y0:  %21.14Le \n\n", Y0target[6] );

       //run iterate for victim satellite
    SimpGenPert4 pertvictim;   // fix to one class variable
       pertvictim.XtoY(X0victim, Y0victim);

       fprintf( pDebug, " Victim Satellite initial equinoctal state" );
       fprintf( pDebug, "\n\n Xo -> Y0:  %21.14Le  %21.14Le  %21.14Le\n",
Y0victim[0], Y0victim[1], Y0victim[2] );
    fprintf( pDebug, " Xo -> Y0:  %21.14Le  %21.14Le  %21.14Le\n", Y0victim[3],
Y0victim[4], Y0victim[5] );
       fprintf( pDebug, " Xo -> Y0:  %21.14Le \n\n", Y0victim[6] );

       // iterate to convergence, or ten iterations, whichever is first
       //Target satellite

       int itert = 0;
       bool convergedt;

       do {

              // increment iteration counter
             itert++;
             fprintf( pDebug, "\n\n Target Iteration %2i\n", itert );

             // calculate one iteration
             iterate(satnot, epocht, Y0target, r0target, v0target, perttarget,
bstart, rtarget, vtarget, Yst, pDebug);

             // print current values
             fprintf(pDebug, "rtarget: %21.14Le %21.14Le %21.14Le \n", rtarget[0],
rtarget[1], rtarget[2]);
             fprintf(pDebug, "vtarget: %21.14Le %21.14Le %21.14Le \n", vtarget[0],
vtarget[1], vtarget[2]);

             // be wildly optimistic...
             convergedt = true;
             for (int i = 0; i < 3; i++) {

                  deltart[i] = r0target[i] - rtarget[i];
                  deltavt[i] = v0target[i] - vtarget[i];
                  if( abs( deltart[i] ) > 1.e-10 ) convergedt = false;
                  if( abs( deltavt[i] ) > 1.e-10 ) convergedt = false;

              }

             // recycle Yst as next guess
             fprintf( pDebug, "\n next state guess Y \n");
             for( int i = 0; i < 7; i++ ) {
                Y0target[i] = Yst[i];
                fprintf( pDebug, " %21.14Le ", Y0target[i] );
                if( i == 2 ) fprintf( pDebug, "\n");
             }
              fprintf( pDebug, "\n");


             // continue as long as less than 10 iterations and we haven't
converged
       } while ( itert < 10 && !convergedt );

        if( itert < 10 ) fprintf( pDebug, "converged");

//Victim satellite

int iterv = 0;
       bool convergedv;

       do {

              // increment iteration counter
             iterv++;
             fprintf( pDebug, "\n\n Victim Iteration %2i\n", iterv );

             // calculate one iteration
             iterate(satnov, epocht, Y0victim, r0victim, v0victim, pertvictim,
bstarv, rvictim, vvictim, Ysv, pDebug);

             // print current values
             fprintf(pDebug, "rvictim: %21.14Le %21.14Le %21.14Le \n", rvictim[0],
rvictim[1], rvictim[2]);
             fprintf(pDebug, "vvictim: %21.14Le %21.14Le %21.14Le \n", vvictim[0],
vvictim[1], vvictim[2]);

             // be wildly optimistic...
             convergedv = true;

             for (int i = 0; i < 3; i++) {

                  deltarv[i] = r0victim[i] - rvictim[i];
                  deltavv[i] = v0victim[i] - vvictim[i];
                  if( abs( deltarv[i] ) > 1.e-10 ) convergedv = false;
                  if( abs( deltavv[i] ) > 1.e-10 ) convergedv = false;
              }

             // recycle Yst as next guess
             fprintf( pDebug, "\n next state guess Y \n");
             for( int i = 0; i < 7; i++ ) {
                Y0victim[i] = Ysv[i];
                fprintf( pDebug, " %21.14Le ", Y0victim[i] );
                if( i == 2 ) fprintf( pDebug, "\n");
             }
              fprintf( pDebug, "\n");


             // continue as long as less than 10 iterations and we haven't 
converged
       } while ( iterv < 10 && !convergedv );

        if( iterv < 10 ) fprintf( pDebug, "converged");

              //now that we have a converged r and v for both the target and 
victim satellites, back the target satellite up to time of maneuver
              //set last iterated vaue for r and v target equal to r and v target 
b to back them up

              double Period = (120*pi)/X0target[0];

              //want to look at different points in the orbit to perform the 
maneuver, these must be in days
              double time[14] = {(Period/4)/86400, (Period/3)/86400, 
(Period/2)/86400, 2*(Period/3)/86400, 3*(Period/4)/86400, 
                     Period/86400,5*(Period/4)/86400, 4*(Period/3)/86400, 
3*(Period/2)/86400, 2*Period/86400, 2.5*Period/86400, 3*Period/86400, 
4*Period/86400, 5.5*Period/86400};
              int arraysize = sizeof(time)/8;

              FILE* Backup;
              Backup = fopen ("TargetBackup.txt", "w");
              FILE* Close;
              Close = fopen ("TargetClose.txt", "w");

              for (int z=0; z<arraysize; z++)
              {
                     fprintf(pDebug, "\n\n time interval %14.7Le days\n",time[z]);
                     fprintf(Probability, "\ntime interval %14.7Le 
days\n",time[z]);
                     fprintf(Probability, " Probability-Uncontrained Constrained 
       Maneuver in km/s\n");
                     fprintf( Constraint, "\ntime interval %14.7Le \n", time[z]);

                     double timeback = time[z]; //how many days prior to 
conjunction do you want to move satellite
                     double backepoch = epocht-timeback;
                     double Xst[7];
```

```
                     perttarget.Solution(backepoch, rtargetb, vtargetb, drvdYtbptr,
Backup);
                     perttarget.YtoX(Yst, Xst);
                     perttarget.UpdateEpoch(epocht, Xst, backepoch, Xtargetb);

                     fprintf( Backup, "\n\n initializing SGP4.  Y:  %21.14Le
%21.14Le  %21.14Le\n", Yst[0], Yst[1], Yst[2] );
                     fprintf( Backup, " initializing SGP4.  Y:  %21.14Le  %21.14Le
%21.14Le\n", Yst[3], Yst[4], Yst[5] );
                     fprintf( Backup, " initializing SGP4.  Y:  %21.14Le \n",
Yst[6] );
                     fprintf( Backup, "\n\n SGP4 call time %21.14Le \n", backepoch
);
                     fprintf( Backup, "number of days prior to conjunction %21.14Le
\n", timeback );
                     fprintf( Backup, " r from Sgp4  %21.14Le  %21.14Le
%21.14Le\n", rtargetb[0], rtargetb[1], rtargetb[2] );
                     fprintf( Backup, " v from Sgp4  %21.14Le  %21.14Le
%21.14Le\n\n", vtargetb[0], vtargetb[1], vtargetb[2] );
                     fprintf( Backup, " XTargetb %21.14Le  %21.14Le  %21.14Le\n",
Xtargetb[0], Xtargetb[1], Xtargetb[2] );
                     fprintf( Backup, " XTargetb %21.14Le  %21.14Le  %21.14Le\n",
Xtargetb[3], Xtargetb[4], Xtargetb[5] );
                     fprintf( Backup, " XTargetb %21.14Le\n", Xtargetb[6] );

       /*now target has been propagated "backwards" to a time prior to the
potential conjunction time*/

       //invert drvdy and reduce to a 6x6 by removing column with airdrag term
       //reduce 6x7 to a 6x6
       //reduce drvdY to a 6x6
              MatDoub matrvYtb(6,6), invdrvdYtb(6,6);
              int x = 0;
              for (int i = 0; i < 6; i++)
              {
                     x = 0; // reset to zero
                     for(int j = 0; j < 7; j++)
                     {
                            if (j != 2)
                            {
                                   matrvYtb[i][x] = drvdYtb[i][j];
                                   x++;
                            }
                     }
              }
 //debug matrvYtb
              fprintf( pDebug, "\n\n 6 x 6 matrvY for partial matrix at tmaneuver
wrt EOE at tclose\n");
              for( int i = 0; i < 6; i++ ) {
                 for( int j = 0; j < 6; j++ ) {
                      fprintf( pDebug, " %14.7Le ", matrvYtb[i][j] );
                 }
                 fprintf( pDebug, "\n");
              }

              // LUdcmp needs to be declared using its constructor
              LUdcmp ludcmptm(matrvYtb);
              ludcmptm.inverse(invdrvdYtb);

              //extract the top right hand corner (matrix B) and the bottom right
hand corner (matrix D)
              for (int i=0; i<3; i++)            //need top three rows
              {
              x = 0;
              for(int j=3; j<6; j++)      // last 3 columns
              {
                     matB[i][x]=invdrvdYtb[i][j];
                     x++;
              }
              }
              //extract the bottom right corner matrix D
              int y=0;

              for (int i=3; i<6; i++)            //need bottom three rows
              {
              x = 0;
              for(int j=3; j<6; j++)      // last 3 columns
              {
                     matD[y][x]=invdrvdYtb[i][j];
                     x++;
              }
                     y++;
              }
       //debug invdrvdytb
              fprintf( pDebug, "\n 6 x 6 invdrydY for partial matrix at tmaneuver
wrt EOE at tclose\n");
              for( int i = 0; i < 6; i++ ) {
                 for( int j = 0; j < 6; j++ ) {
                      fprintf( pDebug, " %14.7Le ", invdrvdYtb[i][j] );
                 }
                 fprintf( pDebug, "\n");
              }

       //extract E F G and H matrices from drvdY at tclose

              perttarget.Solution(epocht,rtarget,vtarget,drvdYtcptr,Close);

       //output to file
              fprintf( Close, "\n\n initializing SGP4.  Y:  %21.14Le  %21.14Le
%21.14Le\n", Yst[0], Yst[1], Yst[2] );
              fprintf( Close, " initializing SGP4.  Y:  %21.14Le  %21.14Le
%21.14Le\n", Yst[3], Yst[4], Yst[5] );
              fprintf( Close, " initializing SGP4.  Y:  %21.14Le \n", Yst[6] );
              fprintf( Close, "\n\n SGP4 call time %21.14Le \n", epocht );
              fprintf( Close, " r from Sgp4  %21.14Le  %21.14Le  %21.14Le\n",
rtarget[0], rtarget[1], rtarget[2] );
              fprintf( Close, " v from Sgp4  %21.14Le  %21.14Le  %21.14Le\n\n",
vtarget[0], vtarget[1], vtarget[2] );
              fprintf( Close, " XTargetClose %21.14Le  %21.14Le  %21.14Le\n",
Xst[0], Xst[1], Xst[2] );
              fprintf( Close, " XTargetClose %21.14Le  %21.14Le  %21.14Le\n",
Xst[3], Xst[4], Xst[5] );
```

```
    fprintf( Close, " XTargetClose %21.14Le\n", Xst[6] );

double matrvYtc[6][6];
    for (int i = 0; i < 6; i++)
    {
        x = 0; // reset to zero
        for(int j = 0; j < 7; j++)
        {
            if (j != 2)
            {
                matrvYtc[i][x] = drvdYtc[i][j];
                x++;
            }
        }
    }

//matrix E
    for (int i=0; i<3; i++)            //need top three rows
    {
    for(int j=0; j<3; j++)      // first 3 columns
    {
        matE[i][j]=matrvYtc[i][j];
    }
    }
//matrix F
    y=0;
    for (int i=0; i<3; i++)            //need top three rows
    {
    x = 0;
    for(int j=3; j<6; j++)      // last 3 columns
    {
        matF[y][x]=matrvYtc[i][j];
        x++;
    }
        y++;
    }
//matrix G
    y=3;
    for (int i=0; i<3; i++)            //need bottom three rows
    {

    for(int j=0; j<3; j++)      // first 3 columns
    {
        matG[i][j]=matrvYtc[y][j];
    }
        y++;
    }
//matrix H
    y=0;
    for (int i=3; i<6; i++)            //need bottom three rows
    {
    x = 0;
    for(int j=3; j<6; j++)      // last 3 columns
    {
        matH[y][x]=matrvYtc[i][j];
        x++;
```

```
              }
                     y++;
              }
       // debug matrvYtc
              fprintf( pDebug, "\n\n 6 x 6 matrvY for partial matrix at tclose wrt 
EOE at tclose\n");
              for( int i = 0; i < 6; i++ ) {
                 for( int j = 0; j < 6; j++ ) {
                      fprintf( pDebug, " %14.7Le ", matrvYtc[i][j] );
                 }
                 fprintf( pDebug, "\n");
              }
double dv[3];
       //create matrix J (EB+FD)
       //initializing matrices to 0
       for(int j = 0; j < 3; j++)
       {
              for(int i = 0; i < 3; i++)
              {
                     matEB[i][j] = 0;
                     matFD[i][j] = 0;
                     matJ[i][j] = 0;
                     SreliJ[i][j] = 0;
                     JTSreliJ[i][j] = 0;
                     invJTSreliJ[i][j] = 0;
                     JTSreli[i][j] = 0;
                     iJTSiJJTSi[i][j] = 0;
                     delv[i] = 0;
                     dv[i] = 0;
                     delvc[i] = 0;
                     iJTSJvtb[i] = 0;
              }
       }
       //multiply matrix E times matrix B
              for (int row = 0; row < 3; row++)
              {
                     for (int col = 0; col < 3; col++)
                     {
                            for (int inner = 0; inner < 3; inner++)
                            {
                                   matEB[row][col] += matE[row][inner] * 
matB[inner][col];
                            }
                     }
              }

       //matrix FD multiply matrix F and matrix D
              for (int row = 0; row < 3; row++)
              {
                     for (int col = 0; col < 3; col++)
                     {
                            for (int inner = 0; inner < 3; inner++)
                            {
                                   matFD[row][col] += matF[row][inner] * 
matD[inner][col];
                            }
                     }
              }

       //Add matrix EB together with FD to create matrix J
              for (int row=0;row<3;row++)
              {
                     for(int col=0;col<3;col++)
                     {
                            matJ[row][col] = matEB[row][col] + matFD[row][col];
                     }
              }
       //debug mat J
              fprintf( pDebug, "\n\n matrix J\n");
              for( int i = 0; i < 3; i++ ) {
                 for( int j = 0; j < 3; j++ ) {
                      fprintf( pDebug, " %14.7Le ", matJ[i][j] );
                 }
                 fprintf( pDebug, "\n");
              }

       //make J transpose for deltav calculation
              for (int row=0; row<3;row++)
              {
                     for (int col=0; col<3; col++)
                     {
                            matJT[col][row] = matJ[row][col];
                     }
              }

       //Compute Srel inverse
              LUdcmp myludcmptm(Srel);
              myludcmptm.inverse(Srelinverse);
       //debug Srel inverse
              fprintf( pDebug, "\n\n matrix Srel inverse\n");
              for( int i = 0; i < 3; i++ ) {
                 for( int j = 0; j < 3; j++ ) {
                      fprintf( pDebug, " %14.7Le ", Srelinverse[i][j] );
                 }
                 fprintf( pDebug, "\n");
              }
       // multiply srelinverse and J
              for (int row = 0; row < 3; row++)
              {
                     for (int col = 0; col < 3; col++)
                     {
                            for (int inner = 0; inner < 3; inner++)
                            {
                                   SreliJ[row][col] += Srelinverse[row][inner] *
matJ[inner][col];
                            }
                     }
              }

       //multiply JT time SreliJ
              for (int row = 0; row < 3; row++)
              {
                     for (int col = 0; col < 3; col++)
                     {
                            for (int inner = 0; inner < 3; inner++)
                            {
                                   JTSreliJ[row][col] += matJT[row][inner] * 
SreliJ[inner][col];
                            }
                     }
              }

       //take the inverse of JTSreliJ
              LUdcmp JTSJludcmp(JTSreliJ);
              JTSJludcmp.inverse(invJTSreliJ);

       //compute JT time Srelinverse
              for (int row = 0; row < 3; row++)
              {
                     for (int col = 0; col < 3; col++)
                     {
                            for (int inner = 0; inner < 3; inner++)
                            {
                                   JTSreli[row][col] += matJT[row][inner] * 
Srelinverse[inner][col];
                            }
                     }
              }

       //compute (JTSreliJ)^-1*JTSreli
              for (int row = 0; row < 3; row++)
              {
                     for (int col = 0; col < 3; col++)
                     {
                            for (int inner = 0; inner < 3; inner++)
                            {
                                   iJTSiJJTSi[row][col] += invJTSreliJ[row][inner] 
* JTSreli[inner][col];
                            }
                     }
              }

       //calculate current miss distance from rtarget and rvictim -- distance 
between two vectors
             for (int i = 0; i < 3; i++)
                {
                  missr[i] = ( rvictim[i] - rtarget[i] );
          }
       //calculate delta v from -(JTSreliJ)^-1*JTSreli*missr--this delta v is the 
one to create the collision
              for (int row = 0; row < 3; row++)
              {
                     for (int col = 0; col < 3; col++)
                     {
                                   delv[row] += iJTSiJJTSi[row][col] * missr[col];
                     }
              }
       //calculate the relative velocity between the target and victim at close
approach time
                for (int i = 0; i < 3; i++)
                {
                  vrel[i] = ( vvictim[i] - vtarget[i] );
          }
       // Calculate invJTSreliJ times velocity of target at maneuver time
              for (int row = 0; row < 3; row++)
              {
                     for (int col = 0; col < 3; col++)
                     {
                                   iJTSJvtb[row] += invJTSreliJ[row][col] *
vtargetb[col];
                     }
              }
              for (int row = 0; row < 3; row++)
              {
                     for (int col = 0; col < 3; col++)
                     {
                                   vrelJ[row] += vrel[row]*matJ[col][row];
                     }
              }
              for (int row = 0; row < 3; row++)
              {
                     for (int col = 0; col < 3; col++)
                     {
                                   iJTSJJvrel[row] += invJTSreliJ[row][col] *
vrelJ[col];
                     }
              }
       double VR = vrel[0]*missr[0]+vrel[1]*missr[1]+vrel[2]*missr[2];
       double VJ1 = vrel[0]*matJ[0][0]+vrel[1]*matJ[1][0]+vrel[2]*matJ[2][0];
       double VJ2 = vrel[0]*matJ[0][1]+vrel[1]*matJ[1][1]+vrel[2]*matJ[2][1];
       double VJ3 = vrel[0]*matJ[0][2]+vrel[1]*matJ[1][2]+vrel[2]*matJ[2][2];
       double B = iJTSJvtb[0];
       double E = iJTSJvtb[1];
       double H = iJTSJvtb[2];
       double A = delv[0];
       double D = delv[1];
       double G = delv[2];
       double C = iJTSJJvrel[0];
       double F = iJTSJJvrel[1];
       double K = iJTSJJvrel[2];

//unconstrained delta v
       double udelv[3];

       udelv[0] = (VJ3*C*G-A*K*VJ3-VR*C+D*VJ2*C-A*F*VJ2)/(VJ1*C+K*VJ3+F*VJ2);
       double lambda = (-2*A-2*udelv[0])/C;
       udelv[1] = -D-0.5*lambda*F;
       udelv[2] = -G-0.5*lambda*K;

//Constrained delta v to maintain orbital period
       double con = -G*vtargetb[2]*B*(B*F-C*E)+A*H*vtargetb[2]*(B*F-
C*E)+B*D*K*B*vtargetb[2]-A*E*K*B*vtargetb[2]-
D*C*H*B*vtargetb[2]+A*E*C*H*vtargetb[2];
```

```
       double con2 = -vtargetb[0]*B*(B*F-C*E)-H*vtargetb[2]*(B*F-
C*E)+K*B*vtargetb[2]-E*C*H*vtargetb[2]+(VJ1*vtargetb[2]-
VJ3*vtarget[0])*(B*K*B*vtargetb[2]+vtargetb[1]*B*(B*F-C*E)-C*H*B*vtargetb[2]);

       double delvxcon = ((VJ2*vtargetb[2]+VJ3*vtargetb[1])*con-
VR*vtargetb[2]*(B*K*B*vtargetb[2]+vtargetb[1]*B*(B*F-C*E)-
C*H*B*vtargetb[2]))/con2;
       double delvycon = (-VR*vtargetb[0]-delvxcon*(VJ1*vtargetb[2]-
VJ3*vtargetb[0]))/(VJ2*vtargetb[2]+VJ3*vtargetb[1]);
       double delvzcon = (-vtargetb[0]*delvxcon-vtargetb[1]*delvycon)/vtargetb[2];
       delvc[0] = delvxcon;
       delvc[1] = delvycon;
       delvc[2] = delvzcon;



//Unconstrained  probability calculations have delv
//Constrained probability calculations have delvc--thei is the change in energy of
the orbit is = 0 (not changing a or Period)
       //want to normalize the calculated delv above--this will give us the
direction to maneuver
       double deltavm[3], deltavmc[3];
       double magdelv = sqrt(udelv[0]*udelv[0] + udelv[1]*udelv[1] +
udelv[2]*udelv[2]);
       double magdelvc = sqrt(delvc[0]*delvc[0] + delvc[1]*delvc[1] +
delvc[2]*delvc[2]);
       double normdelv[3], normdelvc[3];
              for (int i=1; i<3; i++)
              {
                     deltavm[i] = 0;
                     deltavmc[i] = 0;
                     normdelv[i] = 0;
                     normdelvc[i] = 0;
              }
              for (int i=0; i<3; i++)
              {
                     normdelv[i] = udelv[i]/magdelv;
                     normdelvc[i] = delvc[i]/magdelvc;
              }
       //want to multiply the amount of delta v that you wish to expend by the
normdelv direction
               //the delta v of the maneuver to be performed km
       double amount[20] = {0.0, 0.0001, 0.0002, 0.0003, 0.0004, 0.0005, 0.0006,
0.0007, 0.0008, 0.0009, 0.0010, 0.0020, 0.0030, 0.0040,
                     0.0050, 0.0060, 0.0070, 0.0080, 0.0090, 0.0100};

       int array = sizeof(amount)/8;
//Unconstrained outer loop to try varying deltav maneuvers for the same time stamp
of maneuver
       for (int x=0; x<array; x++)
       {
              for (int i=0; i<3; i++)
              {
                     deltavm[i] = -amount[x]*normdelv[i];
              }
       //
```

```
    //Calculate the probability of collision for the given maneuver amount
    for (int i=0; i<3; i++)
    {
        Jdelv[i] = 0;
        delrJdelv[i] = 0;
        SrelidelrJdelv[i] = 0;
    }
    rJvSrJv = 0;
    magSrel = 0;
    coeff = 0;
    for(int i=0; i<array;i++)
    {
            Pcol[i] = 0;
    }

        for (int row = 0; row < 3; row++)
        {
            for (int col = 0; col < 3; col++)
            {
                    Jdelv[row] += matJ[row][col] * deltavm[col];
            }
        }
        for (int i=0; i<3; i++)
        {
            delrJdelv[i] = missr[i] + Jdelv[i];
        }
        for (int row = 0; row < 3; row++)
        {
            for (int col = 0; col < 3; col++)
            {
                    SrelidelrJdelv[row] += Srelinverse[row][col] *
delrJdelv[col];
            }
        }
        rJvSrJv =
(delrJdelv[0]*SrelidelrJdelv[0]+delrJdelv[1]*SrelidelrJdelv[1]+delrJdelv[2]*Srelid
elrJdelv[2])*-0.5;
    //if the covariance matrix is diagonal, the determinant is the
multiplication of the diagonal(must input full determinant for non diagonal)
        magSrel = (pow(Srel[0][0]*Srel[1][1]*Srel[2][2],-0.5));
        coeff = 2*pi;
        Pcol[x] = (1/coeff)*magSrel*exp(rJvSrJv);



        //Calculate the Constrained probability of collision for the given
maneuver amount
        for (int i=0; i<3; i++)
        {
            deltavmc[i] = -amount[x]*normdelvc[i];
        }
        for (int i=0; i<3; i++)
        {
            Jdelvc[i] = 0;
            delrJdelvc[i] = 0;
            SrelidelrJdelvc[i] = 0;
        }
        rJvSrJvc = 0;
        magSrelc = 0;
        coeffc = 0;
        for(int i=0; i<array;i++)
        {
            Pcolc[i] = 0;
        }

        for (int row = 0; row < 3; row++)
        {
            for (int col = 0; col < 3; col++)
            {
                    Jdelvc[row] += matJ[row][col] * deltavmc[col];
            }
        }
        for (int i=0; i<3; i++)
        {
            delrJdelvc[i] = missr[i] + Jdelvc[i];
        }
        for (int row = 0; row < 3; row++)
        {
            for (int col = 0; col < 3; col++)
            {
                    SrelidelrJdelvc[row] += Srelinverse[row][col] *
delrJdelvc[col];
            }
        }
        rJvSrJvc =
(delrJdelvc[0]*SrelidelrJdelvc[0]+delrJdelvc[1]*SrelidelrJdelvc[1]+delrJdelvc[2]*S
relidelrJdelvc[2])*-0.5;
    //if the covariance matrix is diagonal, the determinant is the
multiplication of the diagonal(must input full determinant for non diagonal)
        magSrelc = (pow(Srel[0][0]*Srel[1][1]*Srel[2][2],-0.5));
        coeffc = 2*pi;
        Pcolc[x] = (1/coeffc)*magSrelc*exp(rJvSrJvc);


        fprintf( Constraint, " \n Probability of Collision\n");
        fprintf( Constraint, " %14.7Le\n", Pcolc[x]);
        fprintf( Constraint, "\n\n");
        fprintf( Probability, " %14.7Le %14.7Le %14.7Le", Pcol[x],
Pcolc[x],amount[x]);
        fprintf( Probability, "\n");
    }
    //fprintf(Probability, "\n");

}
}
```

```
void iterate(int satno, double epoch, double *Y, double *r0, double *v0,
SimpGenPert4 &pert, double &bstar,
              double *r, double *v, double *Ys, FILE *pDebug )
{
       //declare variable used within iterate
       double time, deltar[3], deltav[3], deltarv[6], deltaY[6], YSn[7];
       double drvdY[6][7];
       double* drvdYptr = drvdY[0];
       FILE* p9;
       p9 = fopen ("iteratedebug.txt","w");

           fprintf( pDebug, "\n\n initializing SGP4.  Y:  %21.14Le  %21.14Le
%21.14Le\n", Y[0], Y[1], Y[2] );
        fprintf( pDebug, " initializing SGP4.  Y:  %21.14Le  %21.14Le
%21.14Le\n", Y[3], Y[4], Y[5] );
              fprintf( pDebug, " initializing SGP4.  Y:  %21.14Le \n", Y[6] );

              pert.Initialize(satno, epoch, Y);
              time = epoch;
              pert.Solution(time, r, v, drvdYptr, NULL);

              fprintf( pDebug, "\n\n SGP4 call time %21.14Le \n", time );
              fprintf( pDebug, " r from Sgp4  %21.14Le  %21.14Le  %21.14Le\n",
r[0], r[1], r[2] );
           fprintf( pDebug, " v from Sgp4  %21.14Le  %21.14Le  %21.14Le\n\n",
v[0], v[1], v[2] );

              for (int i = 0; i < 3; i++)
              {
                     deltar[i] = r0[i] - r[i];
                     deltav[i] = v0[i] - v[i];
                     fprintf( pDebug, " err r, v:  %14.7Le  %14.7Le\n", deltar[i],
deltav[i] );
              }

              //reduce drvdY to a 6x6
              MatDoub matrvY(6,6), invdrvdY(6,6);
              int x = 0;
              for (int i = 0; i < 6; i++)
              {
                     x = 0; // reset to zero
                     for(int j = 0; j < 7; j++)
                     {
                            if (j != 2)
                            {
                                   matrvY[i][x] = drvdY[i][j];
                                   x++;
                            }
                     }
              }
              // debug
              fprintf( pDebug, "\n 6 x 6 matrvY \n");
              for( int i = 0; i < 6; i++ ) {
                 for( int j = 0; j < 6; j++ ) {
                      fprintf( pDebug, " %14.7Le ", matrvY[i][j] );
                 }
                 fprintf( pDebug, "\n");
              }


              // LUdcmp needs to be declared using its constructor
              LUdcmp myludcmp(matrvY);
              myludcmp.inverse(invdrvdY);

              x = 0;
              for (int i = 0; i < 3; i++)
              {
                     deltarv[x] = deltar[i];
                     deltarv[x + 3] = deltav[i];
                     x++;
              }

              //now you have you inverse drvdY and your delta r and v vectors,
multiply the two together to get your delta Y

              for (int i=0; i<6; i++)
              {
                     deltaY[i] = 0.0; //zero element to be calculated

                     for (int j=0; j<6; j++)
                     {
                            deltaY[i] +=invdrvdY[i][j] * deltarv[j]; //sum directly
into target variable
                     }
                     fprintf( pDebug, " correction %14.7Le\n", deltaY[i] );
              }

              //now need to increment your equinoctal elements by the deltaY you
just calculated
              x = 0;
              for (int i = 0; i < 7; i++)
              {
                     if (i != 2)
                     {
                            Ys[i] = Y[i] + deltaY[x];
                            x++;
                     }
                     else
                     {
                            Ys[i] = bstar;
                     }
              }

}
```

Bibliography

Alfriend, Kyle T, Maruthi R Akella, Joseph Frisbee, James L Foster, Deok-Jin Lee, and Matthew Wilkins. 1999. "Probability of Collision Error Analysis." *Space Debris* 21-35.

Berend, N. 1999. "Estimation of the Probability of Collision between two Catalogued Orbiting Objects." *Advances in Space Research* 23 (1): 243-247.

Chan, Ken. 1997. "Collision Probability Analysis for Earth Orbiting Satellites." *Advances in the Astronautical Sciences* (The Aerospace Corporation) 1033-1048.

DeMars, Kyle J, Yang Cheng, and Moriba K Jah. 2014. "Collision Probability with Gaussian Mixture Orbit Uncertainty." *Journal of Guidance, Control, and Dynamics.*

McKinley, David P. 2006. "Development of a Nonlinear Probability of Collision Tool for the Earth Observing System." *AIAA/AAS Astrodynamics Specialist Conference and Exhibit.* Lanham, MD: American Insitute of Aeronautics and Astronautics.

2012. *NASA Orbital Debris Program Office.* NASA. March. Accessed November 2013. http://orbitaldebris.jsc.nasa.gov/faqs.html#3.

Orbital Debris Quarterly News. 2011. January: 10.

Orbital Debris Quarterly News. 2009. "Satellite Collision Leaves Significant Debris Cloud." April: 1-2.

Patera, Russell P. 2001. "General Method for Calculating Satellite Collision Probability." *Journal of Guidance, Control and Dynamics* 716-722.

Patera, Russell P. 2007. "Space Vehicle Conflict-Avoidance Analysis." *Journal of Guidance, Control, and Dynamics* 30 (2): 492-498.

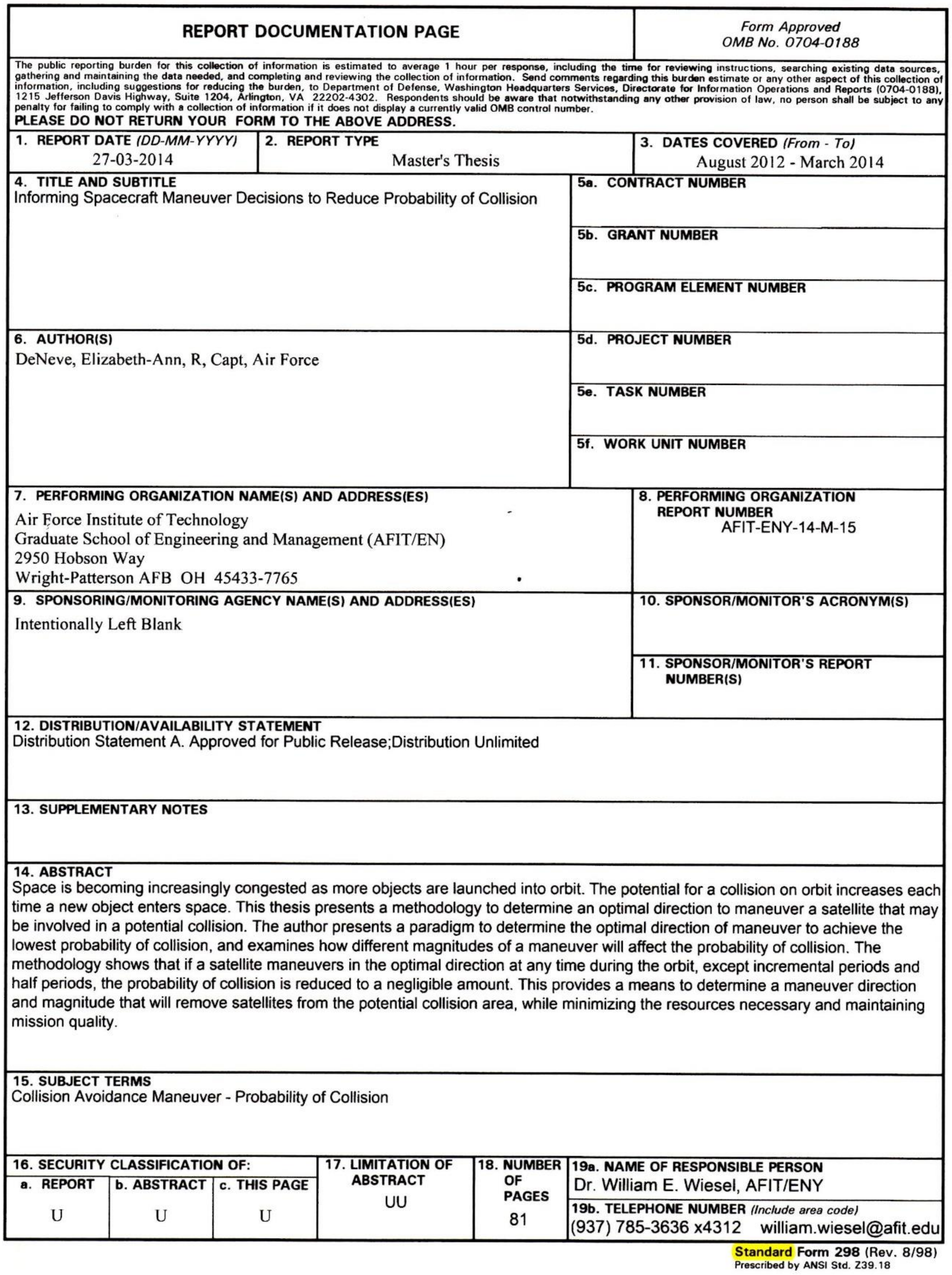
2013. *USSTRATCOM Space Control and Space Surveillance.* Accessed October 16, 2013. http://www.stratcom.mil/factsheets/USSTRATCOM_Space_Control_and_Space_Surveillance/.

Vallado, David A., and Paul Crawford. 2008. "SGP4 Orbit Determination." *AIAA/AAS Astrodynamics Specialist Conference and Exhibit.* Honolulu.

Wertz, James R., David F. Everett, and Jeffery J. Puschell. 2011. *Space Mission Engineering: The New SMAD.* Hawthorne: Microcosm Press.

Wiesel, William E. 2010A. *Modern Orbit Determination.* Beavercreek: Aphelion Press.

—. 2010B. *Spaceflight Dynamics Third Edition.* Beavercreek: Aphelion Press.



REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)* 27-03-2014	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From - To)* August 2012 - March 2014

4. TITLE AND SUBTITLE Informing Spacecraft Maneuver Decisions to Reduce Probability of Collision	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) DeNeve, Elizabeth-Ann, R, Capt, Air Force	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way Wright-Patterson AFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENY-14-M-15
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally Left Blank	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Statement A. Approved for Public Release;Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Space is becoming increasingly congested as more objects are launched into orbit. The potential for a collision on orbit increases each time a new object enters space. This thesis presents a methodology to determine an optimal direction to maneuver a satellite that may be involved in a potential collision. The author presents a paradigm to determine the optimal direction of maneuver to achieve the lowest probability of collision, and examines how different magnitudes of a maneuver will affect the probability of collision. The methodology shows that if a satellite maneuvers in the optimal direction at any time during the orbit, except incremental periods and half periods, the probability of collision is reduced to a negligible amount. This provides a means to determine a maneuver direction and magnitude that will remove satellites from the potential collision area, while minimizing the resources necessary and maintaining mission quality.

15. SUBJECT TERMS
Collision Avoidance Maneuver - Probability of Collision

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. William E. Wiesel, AFIT/ENY
U	U	U	UU	81	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 785-3636 x4312 william.wiesel@afit.edu

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18